(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **06274384 A**

(43) Date of publication of application: **30.09.94**

(51) Int. Cl **G06F 12/00**

(21) Application number: **05062181**

(22) Date of filing: **22.03.93**

(71) Applicant: **N T T DATA TSUSHIN KK**

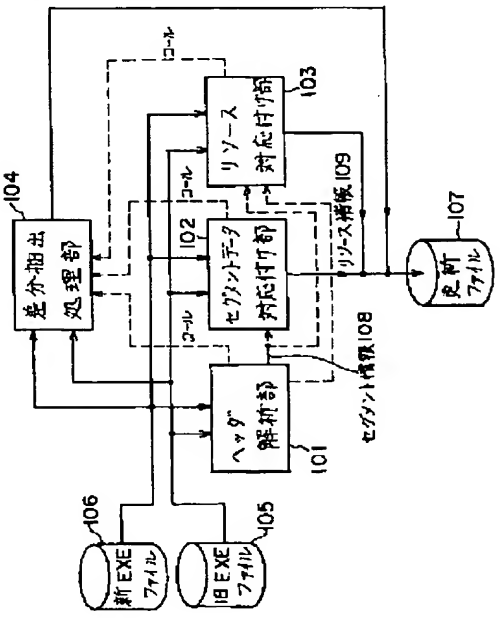
(72) Inventor: **TANIMURA MORIMASA**

(54) **EXTRACTING/UPDATING DEVICE OF EXECUTION FILE DIFFERENCE AND EXTRACTING METHOD FOR EXECUTION FILE DIFFERENCE**

(57) Abstract:

PURPOSE: To reduce the size of an update file storing difference information to the old execution file of a new execution file.

CONSTITUTION: Concerning respective old and new EXE files 105 and 106, a header analysis part 101 extracts the areas of their headers and the areas of respective tables and calls a difference extraction processing part 104 concerning these areas. A segment data coordination part 102 coordinates segment data between the files 105 and 106 and calls the difference extraction processing part 104 concerning the coordinated area. A resource coordination part 103 coordinates a resource between the files 105 and 106 and calls the difference extraction processing part 104 concerning the coordinated area. A difference extraction processing part 104 extracts the difference of a byte unit to the file 105 of the file 106 from the top and the end of the areas specified from the respective parts 101 to 103 and saves the difference in the update file 107.



(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平6－274384

(43)公開日　平成６年(1994)９月30日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ Ｉ	技術表示箇所
Ｇ０６Ｆ　12/00	５１０　Ｂ	8944－5B		

審査請求　未請求　請求項の数６　ＯＬ　（全 30 頁）

(21)出願番号　特願平5－62181

(22)出願日　平成５年(1993)３月22日

(71)出願人　000102728
エヌ・ティ・ティ・データ通信株式会社
東京都江東区豊洲三丁目３番３号

(72)発明者　谷村　守正
東京都江東区豊洲三丁目３番３号　エヌ・ティ・ティ・データ通信株式会社内

(74)代理人　弁理士　大菅　義之

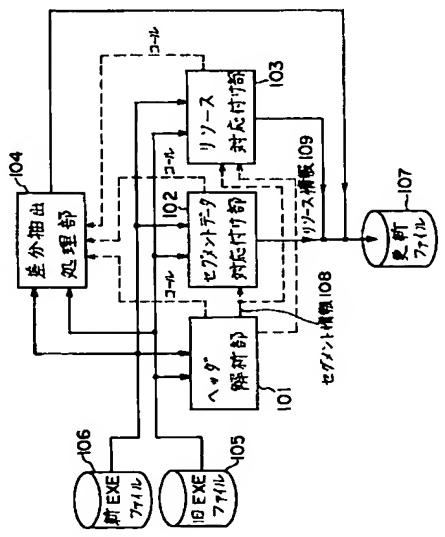
(54)【発明の名称】　実行ファイル差分抽出／更新装置及び実行ファイル差分抽出方法

(57)【要約】

【目的】　本発明は、新実行ファイルの旧実行ファイルに対する差分情報を格納する更新ファイルのサイズを削減することを目的とする。

【構成】　ヘッダ解析部１０１は、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき、そのヘッダの領域及び各テーブル部の領域を抽出し、それらの領域につき差分抽出処理部１０４をコールする。セグメントデータ対応付け部１０２は、ファイル１０５と１０６との間のセグメントデータの対応付けを行い、対応付けされた領域について差分抽出処理部１０４をコールする。リソース対応付け部１０３は、ファイル１０５と１０６との間のリソースの対応付けを行い、対応付けされた領域について差分抽出処理部１０４をコールする。差分抽出処理部１０４は、１０１、１０２、又は１０３の各部から指定された領域の先頭から末尾まで、ファイル１０６の１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

本発明の実施例の構成図



1

【特許請求の範囲】

【請求項１】　旧実行ファイルを新実行ファイルに更新するための更新ファイルを、新実行ファイルの旧実行ファイルに対する差分として抽出する実行ファイル差分抽出装置において、

前記旧実行ファイル及び前記新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割する領域分割手段と、

該領域分割手段によって分割された前記旧実行ファイルの複数の領域と前記新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けする領域対応付け手段と、

該領域対応付け手段によって対応付けされた前記旧実行ファイルの領域と前記新実行ファイルの領域との間で、前記旧実行ファイルの領域のデータに対する前記新実行ファイルの領域のデータの差分情報を抽出する差分抽出処理手段と、

該差分抽出処理手段において抽出された差分情報を前記更新ファイルに格納すると共に、前記領域対応付け手段において対応付けが行われなかった前記旧実行ファイルの領域又は前記新実行ファイルの領域に対応する差分情報を抽出しそれを前記更新ファイルに格納する更新ファイル生成手段と、

を有することを特徴とする実行ファイル差分抽出装置。

【請求項２】　新実行ファイルの旧実行ファイルに対する差分が抽出されて更新ファイルとされ、旧実行ファイルと更新ファイルとから新実行ファイルが生成される実行ファイル更新装置において、

請求項１に記載の更新ファイル生成手段により生成された更新ファイルから前記差分情報を順次取り出し、該差分情報と前記旧実行ファイルとから、前記新実行ファイルを生成するファイル更新手段を有する、

ことを特徴とする実行ファイル更新装置。

【請求項３】　新実行ファイルの旧実行ファイルに対する差分が抽出されて更新ファイルとされ、旧実行ファイルと更新ファイルとから新実行ファイルが生成される実行ファイル差分抽出／更新装置において、

前記旧実行ファイル及び前記新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割する領域分割手段と、

該領域分割手段によって分割された前記旧実行ファイルの複数の領域と前記新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けする領域対応付け手段と、

該領域対応付け手段によって対応付けされた前記旧実行ファイルの領域と前記新実行ファイルの領域との間で、前記旧実行ファイルの領域のデータに対する前記新実行ファイルの領域のデータの差分情報を抽出する差分抽出処理手段と、

該差分抽出処理手段において抽出された差分情報を前記

2

更新ファイルに格納すると共に、前記領域対応付け手段において対応付けが行われなかった前記旧実行ファイルの領域又は前記新実行ファイルの領域に対応する差分情報を抽出しそれを前記更新ファイルに格納する更新ファイル生成手段と、

該更新ファイル生成手段により生成された更新ファイルから前記差分情報を順次取り出し、該差分情報と前記旧実行ファイルとから、前記新実行ファイルを生成するファイル更新手段と、

を有することを特徴とする実行ファイル差分抽出／更新装置。

【請求項４】　旧実行ファイルを新実行ファイルに更新するための更新ファイルを、新実行ファイルの旧実行ファイルに対する差分として抽出する実行ファイル差分抽出方法において、

前記旧実行ファイル及び前記新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割し、

該領域の分割によって分割された前記旧実行ファイルの複数の領域と前記新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けし、

該領域の対応付けによって対応付けされた前記旧実行ファイルの領域と前記新実行ファイルの領域との間で、前記旧実行ファイルの領域のデータに対する前記新実行ファイルの領域のデータの差分情報を抽出し、

該抽出された差分情報を前記更新ファイルに格納すると共に、前記領域の対応付けにおいて対応付けが行われなかった前記旧実行ファイルの領域又は前記新実行ファイルの領域に対応する差分情報を抽出しそれを前記更新ファイルに格納する、

ことを特徴とする実行ファイル差分抽出方法。

【請求項５】　前記論理単位は、前記旧実行ファイル又は前記新実行ファイルのデータ属性の単位であるセグメント、リソース、ヘッダ、又はテーブルである、

ことを特徴とする請求項１乃至４の何れか１項に記載の実行ファイル差分抽出／更新装置又は実行ファイル差分抽出方法。

【請求項６】　前記差分情報は、前記新実行ファイルにおいて前記旧実行ファイルに対してデータの挿入、削除、又は置換が行われる旨の属性情報、その対象となるデータの長さ情報、及びその対象となるデータとから構成される、

ことを特徴とする請求項１乃至５の何れか１項に記載の実行ファイル差分抽出／更新装置又は実行ファイル差分抽出方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、バグの修正又は部分的な機能拡張を目的として実行ファイルが更新される場合に、新実行ファイルの旧実行ファイルに対する差分が抽出されて更新ファイルとされ、旧実行ファイルと更新フ

ファイルとから新実行ファイルが生成される実行ファイル差分抽出／更新装置及び方法に関する。

【０００２】

【従来の技術】ユーザに実行ファイルが配付されその実行ファイル（ＥＸＥファイル）により所定サービスが提供されている場合、その実行ファイルに存在するバグを修正したり、その実行ファイルによって実現される機能の部分的な拡張を目的として、図２８(a) に示されるように、ユーザに配付されている実行ファイルにおいて、新ＥＸＥファイルから旧ＥＸＥファイルへの更新がなされる場合がある。

【０００３】一方、最近では、コンピュータシステムの機能の発達により、実行ファイルのプログラムサイズが数百キロバイトから数メガバイト或いは数十メガバイトに達しつつある。従って、上述のような実行ファイルの更新時に、新ＥＸＥファイルがそのまま配付されるのは、配付媒体に多くのデータ容量が要求されることになり、不経済である。

【０００４】例えば、配付媒体が図２９に示されるようなネットワークである場合、センタＴＣのホストコンピュータＨＯＳＴは、回線を介して、事務所ＴＣ内のホストコンピュータＨＯＳＴに対して新ＥＸＥファイルを配信することになるが、新ＥＸＥファイルが上述のような大きなサイズを有する場合、回線使用料が高くついてしまい、不経済である。

【０００５】このような問題に対し、従来、新ＥＸＥファイルそのものが配付されるのではなく、図２８(b) に示されるように、旧ＥＸＥファイルと新ＥＸＥファイルとの間で比較処理がなされ、新ＥＸＥファイルの旧ＥＸＥファイルに対する差分が抽出されて更新ファイルとされ、その更新ファイルがユーザに配付されるという形態が採られるようになってきている。この場合、ユーザ側では、図２８(b) に示されるように、所定の更新プログラムによる更新処理によって、旧ＥＸＥファイルと更新ファイルとから新ＥＸＥファイルが生成される。新ＥＸＥファイルと旧ＥＸＥファイルとでは、プログラムコードが同じ部分が存在する場合が多いために、それらの差分である更新ファイルのサイズは、新ＥＸＥファイルのサイズに比較して小さい場合が多い。従って、更新ファイルがユーザに配付されることにより、配付媒体に要求されるデータ容量を削減することができる。

【０００６】図２９の例においては、センタＴＣのホストコンピュータＨＯＳＴは、回線を介して、事務所ＴＣ内のホストコンピュータＨＯＳＴに対して更新ファイルを配信する。事務所ＴＣ内のホストコンピュータＨＯＳＴでは、センタ間通信部が、上記更新ファイルを受信した後、更新プログラムを起動することにより、旧ＥＸＥファイルと更新ファイルとから新ＥＸＥファイルを生成する。この新ＥＸＥファイルは、ファイル操作部を介して正ディスク及び副ディスクに保存される。そして、事

務所ＴＣのホストコンピュータＨＯＳＴ内の配信処理管理部が、新ＥＸＥファイルを、事務所内通信部からＬＡＮ（ローカルエリアネットワーク）を介して、クライアントサーバＣＳＶに配信する。クライアントサーバＣＳＶでは、事務所内通信部が上記新ＥＸＥファイルを受信する。この受信された新ＥＸＥファイルは、ファイル操作部を介して正ディスク及び副ディスクに保存される。

【０００７】

【発明が解決しようとする課題】しかし、上述のような従来の差分抽出方式では、旧ＥＸＥファイルの先頭及び新ＥＸＥファイルの先頭から各ファイルの末尾までのファイル全体にわたって、バイト単位でデータが一致するか否かが判定され、一致しないデータが差分のデータとして抽出されている。このために、例えば新ＥＸＥファイルの途中で１バイトでもデータの削除又は挿入が行われていると、そのバイト以降では全て旧ＥＸＥファイルのデータと新ＥＸＥファイルのデータとが一致しなくなり、その結果、更新ファイルのサイズが大きくなってしまうという問題点を有している。従って、図２９の例のようにネットワークを介して更新ファイルが配信される場合などで、新ＥＸＥファイルがそのまま配信される場合に比較してそれほど回線の使用効率が向上しないという結果になってしまう。

【０００８】本発明は、更新ファイルのサイズを削減することを目的とする。

【０００９】

【課題を解決するための手段】本発明は、まず、実行ファイル差分抽出装置として、以下の構成を有する。領域分割手段は、旧実行ファイル及び新実行ファイルのそれぞれを、所定の論理単位毎に複数の領域に分割する。この論理単位は、例えば、旧実行ファイル又は新実行ファイルのデータ属性の単位であるセグメント、リソース、ヘッダ、又はテーブルである。

【００１０】次に、領域対応付け手段は、領域分割手段によって分割された旧実行ファイルの複数の領域と新実行ファイルの複数の領域との間で、互いに対応する領域同士を対応付けする。

【００１１】また、差分抽出処理手段は、領域対応付け手段によって対応付けされた旧実行ファイルの領域と新実行ファイルの領域との間で、旧実行ファイルの領域のデータに対する新実行ファイルの領域のデータの差分情報を抽出する。この差分情報は、例えば、新実行ファイルにおいて旧実行ファイルに対してデータの挿入、削除、又は置換が行われる旨の属性情報、その対象となるデータの長さ情報、及びその対象となるデータとから構成される。

【００１２】そして、更新ファイル生成手段は、差分抽出処理手段において抽出された差分情報を更新ファイルに格納すると共に、領域対応付け手段において対応付けが行われなかった旧実行ファイルの領域又は新実行ファ

イルの領域に対応する差分情報を抽出しそれを更新ファイルに格納する。

【００１３】また、本発明は、実行ファイル更新装置として、上述の更新ファイル生成手段により生成された更新ファイルから前記差分情報を順次取り出し、その差分情報と旧実行ファイルとから、新実行ファイルを生成するファイル更新手段を有するように構成される。

【００１４】

【作用】本発明においては、差分抽出処理が実行される場合、例えばOS/2システムにおける実行ファイルでは、DOS EXE 形式の領域、OS/2 EXEヘッダ領域、各テーブル部領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行される。

【００１５】このため、旧実行ファイルと新実行ファイルとの間で、各領域のデータが一致する確率が高くなり、その結果、生成される更新ファイルのサイズが大幅に削減される。

【００１６】

【実施例】以下、図面を参照しながら本発明の実施例につき詳細に説明する。図１は、本発明の実施例の構成図である。

【００１７】まず、差分抽出処理部１０４は、後述するヘッダ解析部１０１、セグメントデータ対応付け部１０２、又はリソース対応付け部１０３から指定された領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００１８】ヘッダ解析部１０１は、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、そのDOS EXE ヘッダの領域及びOS/2 EXEヘッダの領域を抽出し、それらの領域について差分抽出処理部１０４に対して差分抽出処理を指示する。また、ヘッダ解析部１０１は、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、上述の抽出されたDOS EXE ヘッダ及びOS/2 EXEヘッダの内容を解析することにより後述する各テーブル部の領域を抽出し、抽出された各テーブル部領域についても差分抽出処理部１０４に対して差分抽出処理を指示する。更に、ヘッダ解析部１０１は、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、抽出されたセグメントテーブル部から後述するセグメント情報１０８を抽出して、それを後述するセグメントデータ対応付け部１０２及びリソース対応付け部１０３に設定し、また、抽出されたリソーステーブル部から後述するリソース情報１０９を抽出して、それを後述するリソース対応付け部１０３に設定する。

【００１９】セグメントデータ対応付け部１０２は、ヘッダ解析部１０１からのセグメント情報１０８に基づいて、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との間のセグメントデータの対応付けを行い、対応付けされた各セグメントデータのペアの領域について差分抽出処理部１０４に対して差分抽出処理を指示する。旧ＥＸＥファイル１０５に存在するセグメントデータに対応するものが新ＥＸＥファイル１０６に存在しない場合は、そのセグメントデータを削除すべき旨の差分情報を更新ファイル１０７にセーブする。逆に、新ＥＸＥファイル１０６に存在するセグメントデータに対応するものが旧ＥＸＥファイル１０５に存在しない場合は、そのセグメントデータを挿入すべき旨の差分情報を更新ファイル１０７にセーブする。

【００２０】リソース対応付け部１０３は、ヘッダ解析部１０１からのセグメント情報１０８及びリソース情報１０９に基づいて、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との間のリソースの対応付けを行い、対応付けされた各リソースのペアの領域について差分抽出処理部１０４に対して差分抽出処理を指示する。旧ＥＸＥファイル１０５に存在するリソースに対応するものが新ＥＸＥファイル１０６に存在しない場合は、そのリソースを削除すべき旨の差分情報を更新ファイル１０７にセーブする。逆に、新ＥＸＥファイル１０６に存在するリソースに対応するものが旧ＥＸＥファイル１０５に存在しない場合は、そのリソースを挿入すべき旨の差分情報を更新ファイル１０７にセーブする。

【００２１】上述の構成を有する実施例の動作について、以下に順次説明する。まず、図２は、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６の共通のファイル形式を示した図である。

【００２２】本実施例で対象とされるファイルは、コンピュータのためのマルチタスクオペレーティングシステムの１つであるOS/2システムのもとで実行可能なＥＸＥファイルである。

【００２３】このＥＸＥファイルは、コンピュータのための従来からあるシングルタスクオペレーティングシステムの１つであるMS-DOSシステムのもとでも実行可能なように構築されている。そのため、このＥＸＥファイルは、図２に示されるように、DOS EXE 形式の領域とOS/2 EXE形式の領域とから構成されている。

【００２４】まず、ファイルの前半部分には、DOS EXE 形式の領域が存在する。DOS EXE 形式の領域は、MS-DOS用プログラムを実行するための各種情報と後述するOS/2 EXEヘッダへのポインタ情報等が格納されるDOS EXE ヘッダ領域と、MS-DOSシステムのもとで動作するプログラムの本体が格納されるDOS ロードモジュール領域とから構成される。

【００２５】次に、ファイルの後半部分には、OS/2 EXE 形式の領域が存在する。OS/2 EXE形式の領域は、OS/2用プログラムを実行するための各種情報と後述する各種テーブル部へのポインタ情報等が格納されるOS/2 EXEヘッダ領域、後述する各種テーブル部領域、OS/2システムのもとで動作するプログラムの本体が格納されるセグメン

トデータ領域、及びOS/2プログラムで利用されるリソースと呼ばれるデータが格納されるリソース領域とから構成される。

【００２６】本実施例においては、差分抽出処理が実行される場合、DOS EXE 形式の領域、OS/2 EXEヘッダ領域、及び各テーブル部領域のそれぞれの領域毎に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との差分抽出処理が実行され、その後、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との間のセグメントデータ及びリソースの対応付けが行われた上で、その対応付けされた各セグメントデータ及びリソースのペアの領域毎に差分抽出処理が実行されることが、大きな特徴である。このように、所定の論理単位毎に差分抽出処理が実行されることにより、従来技術に比較して、生成される更新ファイル１０７のサイズを大幅に削減することが可能となる。

【００２７】次に、ヘッダ解析部１０１、セグメントデータ対応付け部１０２、リソース対応付け部１０３、及び差分抽出処理部１０４の各部の動作について、順次説明する。

【００２８】本実施例では、まず、ヘッダ解析部１０１が起動される。ヘッダ解析部１０１で実行される処理の動作フローチャートは、図３及び図４に示される。このヘッダ解析部１０１の動作は、特には図示しないコンピュータシステムにおいて、ＣＰＵ（中央処理装置）が、ディスク装置などに記憶された旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

【００２９】まず、ステップＳ３０１で、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット０に格納されている１ワードのコードが“ＭＺ”であるか否かが判定される（図５参照）。そのコードが“ＭＺ”でない場合、そのファイルはＥＸＥファイルではないため、ステップＳ３２３でエラー表示が行われ差分抽出処理を終了する。

【００３０】上述のコードが“ＭＺ”である場合、ステップＳ３０２で、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット１８ｈ（“ｈ”は、それが付加された１ワードの値が１６進数であることを示す。）に格納されている値が４０ｈであるか否かが判定される（図５参照）。その値が４０ｈでない場合は、そのファイルはOS/2 EXE形式のファイルではないため、ステップＳ３２３でエラー表示が行われ差分抽出処理を終了する。

【００３１】上述の値が４０ｈである場合は、ステップＳ３０３で、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６のそれぞれについて、DOS EXE ヘッダ内のファイルオフセット３Ｃｈに格納されている値で示される１ワードのオフセットアドレスがOS/2 EXEの領域のアドレスであるか否かが判定される（図５参照）。そのアドレスがOS/2 EXEの領域のアドレスでない場合は、そのファイルはOS/2 EXE形式のファイルではないため、ステップＳ３２３でエラー表示が行われ差分抽出処理を終了する。

【００３２】上述のアドレスがOS/2 EXEの領域のアドレスである場合は、更にステップＳ３０４で、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６のそれぞれについて、OS/2 EXEヘッダ内のOS/2 EXE相対オフセット（OS/2 EXE形式の領域の先頭のアドレスが０とされる場合の相対オフセット）の３６ｈに格納されている１ワードの値が１であるか否かが判定される（図５参照）。その値が１でない場合は、そのファイルはOS/2 EXE形式のファイルではないため、ステップＳ３２３でエラー表示が行われ差分抽出処理を終了する。

【００３３】上述の値が１である場合、ステップＳ３０５で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき、DOS EXE 形式のデータ領域が、ファイルオフセット０からDOS EXE ヘッダ内のファイルオフセット３Ｃｈの内容として示されるOS/2 EXEの領域のアドレスの手前のアドレスまでの領域として抽出される（図２参照）。

【００３４】次に、ヘッダ解析部１０１は、ステップＳ３０６で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき抽出されたDOS EXE 形式のデータ領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理により、ヘッダ解析部１０１から渡されたDOS EXE 形式のデータ領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００３５】次に、ステップＳ３０７で、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６における図５に示されるOS/2 EXEヘッダの内容のうち、●印が付与されたデータが抽出され、それぞれ図６に示される各変数に格納される。この変数は、特には図示しないメモリに保持される。これらの変数については後述する。

【００３６】続いて、ステップＳ３０８で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、OS/2 EXEヘッダ領域が、前述したDOS EXE ヘッダ内のファイルオフセット３Ｃｈの内容として示されるOS/2 EXEの領域のアドレスから変数segtbl off（図６参照）で示されるセグメントテーブル部の先頭アドレスの手前のアドレスまでの領域として抽出される（図２参照）。

【００３７】次に、ヘッダ解析部１０１は、ステップＳ３０９で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそ

れぞれについて抽出されたOS/2 EXEヘッダ領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理により、ヘッダ解析部１０１から渡されたOS/2 EXEヘッダ領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００３８】続いて、ステップＳ３１０で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、変数segtbl off及びsegtbl cnt（図６参照）に基づいて抽出されるセグメントテーブル部（図２参照）が解析される。セグメントテーブル部のデータ構造は、図７に示される。セグメントテーブル部は、セグメントデータの先頭アドレスを示す１ワードのファイルオフセットデータ、セグメントデータの実サイズを示す１ワードの実セグメントデータサイズデータ、及びセグメントデータのタイプ及び属性を示す１ワードのセグメントタイプ／属性データからなるデータセットを、後述するセグメントデータのそれぞれに対応して有している。ステップＳ３１０では、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、このデータ構造が例えばデータ構造体の形式で、セグメント情報１０８（図１参照）として、後述するセグメントデータ対応付け部１０２及びリソース対応付け部１０３からアクセス可能な特には図示しないメモリに設定される。

【００３９】次に、ステップＳ３１１で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、上述のセグメントテーブル部領域（図２参照）が抽出される。この領域は、変数segtbl offで示されるセグメントテーブル部の先頭のオフセットアドレスから変数entrytbl offで示されるエキスポートエントリテーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図６、図２参照）。

【００４０】続いて、ヘッダ解析部１０１は、ステップＳ３１２で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて抽出されたセグメントテーブル部領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理により、ヘッダ解析部１０１から渡されたセグメントテーブル部領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００４１】次に、ステップＳ３１３で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき、エキスポートエントリテーブル部領域（図２参照）が抽出される。この領域は、変数entrytbl offで示されるエキスポートエントリテーブル部の先頭のオフセットアドレスから変数resnamtbl off で示される常駐ネームテーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図６、図２参照）。

【００４２】エキスポートエントリテーブル部には、それが含まれるOS/2 EXE形式のプログラムが、他のOS/2プログラムから参照され得るプログラム、例えばＤＬＬ（ダイナミックリンクライブラリ）プログラムである場合に、他のOS/2プログラムがこのテーブル部が含まれるプログラムの一部として提供される関数を参照できるように、その関数へのファイル内のセグメント番号とオフセットが格納される。

【００４３】続いて、ヘッダ解析部１０１は、ステップＳ３１４で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて抽出されたエキスポートエントリテーブル部領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理により、ヘッダ解析部１０１から渡されたセグメントテーブル部領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００４４】次に、ステップＳ３１５で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、常駐ネームテーブル部及び非常駐ネームテーブル部（図２参照）の領域が、１つの領域として抽出される。この領域は、変数resnamtbloff で示される常駐ネームテーブル部の先頭のオフセットアドレスから変数modref offで示されるモジュール参照テーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図６、図２参照）。

【００４５】常駐ネームテーブル部及び非常駐ネームテーブル部は、図８に示される共通の構成を有し、常駐ネームテーブル部には、プログラム実行時にメモリに常駐するモジュール名とそのバイト数及びエントリされるテーブルのインデクスが格納され、非常駐ネームテーブル部には、プログラム実行時に必要に応じてそのプログラム自体が呼出すモジュール名とそのバイト数及びエントリされるテーブル（モユール）のインデクスが格納される。

【００４６】続いて、ヘッダ解析部１０１は、ステップＳ３１６で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて抽出された常駐／非常駐ネームテーブル部領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理によって、ヘッダ解析部１０１から渡された常駐／非常駐ネームテーブル部領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００４７】次に、ステップＳ３１７において、旧ＥＸ

Ｅファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、モジュール参照テーブル部及びインポートネームテーブル部（図２参照）の領域が、１つの領域として抽出される。この領域は、変数modref offで示されるモジュール参照テーブル部の先頭のオフセットアドレスから変数resrctbl offで示されるリソーステーブル部の先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される（図６、図２参照）。

【００４８】インポートネームテーブル部には、図９に示されるように、それが含まれるプログラムが参照する他のＤＬＬプログラムのモジュール名とそのバイト長が格納され、モジュール参照テーブル部には、図１０に示されるように、インポートネームテーブル部の各要素へのオフセット値がワード配列として格納される。

【００４９】続いて、ヘッダ解析部１０１は、ステップＳ３１８で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて抽出されたモジュール参照／インポートネームテーブル部領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理によって、ヘッダ解析部１０１から渡されたモジュール参照／インポートネームテーブル部領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００５０】次に、ステップＳ３１９で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、変数resrctbl off及びresrctbl cnt（図６参照）に基づいて抽出されるリソーステーブル部（図２参照）が解析される。リソーステーブル部のデータ構造は、図１１に示される。リソーステーブル部は、リソースのタイプの識別子を示す１ワードのリソースタイプＩＤデータと、リソースネームの識別子を示す１ワードのリソースネームＩＤデータとからなるデータセットを、後述するリソースのそれぞれに対応して有している。ステップＳ３１９では、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、このデータ構造が、リソース情報１０９（図１参照）として、後述するリソース対応付け部１０３からアクセス可能な特には図示しないメモリに設定される。

【００５１】次に、ステップＳ３２０で、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて、上述のリソーステーブル部領域（図２参照）が抽出される。この領域は、変数resrctbl offで示されるリソーステーブル部の先頭のオフセットアドレスから変数resrctbl cntで示されるリソースエントリ数分だけ進んだアドレスまでの領域として抽出される（図６、図２参照）。

【００５２】続いて、ヘッダ解析部１０１は、ステップＳ３２１で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき抽出されたリソーステーブル部領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理により、ヘッダ解析部１０１から渡されたリソーステーブル部領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００５３】以上のようにして、ヘッダ解析部１０１によって、DOS EXE 形式の領域、OS/2EXEヘッダ領域、及び各テーブル部領域のそれぞれの領域毎に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との差分抽出処理が実行される。

【００５４】そして、ヘッダ解析部１０１は、ステップＳ３２２で、図１のセグメントデータ対応付け部１０２を起動して、処理を終了する。セグメントデータ対応付け部１０２で実行される処理の動作フローチャートは図１２に示される。このセグメントデータ対応付け部１０２の動作は、ヘッダ解析部１０１の場合と同様、特には図示しないコンピュータシステムにおいて、ＣＰＵが、ディスク装置などに記憶された旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

【００５５】ここで、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６とで対応するセグメントデータがある場合、変更されていないセグメントについてはそれらの内容が一致し、変更されたセグメントについてのみ差分が生ずる。そこで、セグメントデータ対応付け部１０２では、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との間で、セグメントデータの属性やその主要部のデータを比較することによりその対応付けを行い、その対応付けしたセグメントデータ領域毎に差分抽出処理部１０４に対して差分抽出処理を指示する。

【００５６】まず、ステップＳ１２０１で、前述した図４のステップＳ３１０でヘッダ解析部１０１から設定されたセグメント情報１０８に基づいて、セグメントの対応付けが行われる。

【００５７】この処理では、まず、旧ＥＸＥファイル１０５のセグメント情報１０８（図７参照）に基づいて、旧ＥＸＥファイル１０５のセグメントが一列に並べられる。なお、セグメントは、リンカによってファイルオフセットが増大する方向に並べられている。並べられた旧ＥＸＥファイル１０５のセグメントの例を図１３(a)に示す。

【００５８】次に、新ＥＸＥファイル１０６のセグメント情報１０８（図７参照）に基づいて、新ＥＸＥファイル１０６のセグメントが一列に並べられる。並べられた新ＥＸＥファイル１０６のセグメントの例を図１３(b)に示す。

【００５９】図１３の例では、新ＥＸＥファイル１０６においては、旧ＥＸＥファイル１０５に対して、セグメント番号３のセグメントが追加されている。従って、この状態が検出される必要がある。

【００６０】今、本実施例では、旧ＥＸＥファイル１０５から新ＥＸＥファイル１０６への更新は、セグメントを削除、置換、又は追加することにより行われるとする。従って、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のセグメントの対応関係が図１４(a) に示されるようなものになることはあり得ず、その対応関係は例えば図１４(b) に示される如くとなる。この例では、旧ＥＸＥファイル１０５のセグメント番号４のセグメントが削除され、新ＥＸＥファイル１０６のセグメント番号２のセグメントが追加され、その他のセグメントのペアについては、変更がないか、又は置換がなされている。

【００６１】そこで、一列に並べられた旧ＥＸＥファイル１０５のセグメント及び新ＥＸＥファイル１０６のセグメントについて、以下の対応付け処理がなされる。

処理１：旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のセグメント数が比較される。

【００６２】処理２：セグメント数が同じか、新ＥＸＥファイル１０６のセグメント数の方が多い場合は、旧ＥＸＥファイル１０５で定義されているセグメントをキーとして、新ＥＸＥファイル１０６中のセグメントがサーチされる。旧ＥＸＥファイル１０５のセグメント数の方が多い場合は、新ＥＸＥファイル１０６で定義されているセグメントをキーとして、旧ＥＸＥファイル１０５中のセグメントがサーチされる。

【００６３】処理３：サーチ対象であるセグメントにおいて、現在のキーとされているセグメントに対し、セグメントタイプ／属性が同じで、かつ、セグメントデータの最初の１６バイトの内容が一致するものがペアセグメントとされる。ここで、図１５に示されるように、先頭にリロケーション情報が設定されているリロケーションテーブルを有するセグメントデータが存在する。キーとされているセグメント又はサーチ対象であるセグメントの何れかがこのようなデータである場合には、このリロケーションテーブルの直後の最初の１６バイトの内容が一致するものがペアセグメントとされ、リロケーションテーブルについては、後述するように別に差分抽出処理が実行される。なお、リロケーションテーブルが存在するか否かは、セグメント情報１０８のセグメントタイプ／属性データに、リロケーションテーブルが存在することを示すデータ値が設定されているか否かによって判別できる。

【００６４】サーチ対象であるセグメントにおいて、上記一致関係が成立するセグメントがない場合には、キーとなったセグメントについては、サーチは諦められ、そのセグメントは旧ＥＸＥファイル１０５の孤立セグメントとされる。そして、次のセグメントがキーとされて、サーチが行われる。

【００６５】全てのサーチが終了した時点で、新ＥＸＥファイル１０６においてペアとならなかったセグメントは、新ＥＸＥファイル１０６の孤立セグメントとされる。このようにして得られた、ペアセグメント、旧ＥＸＥファイル１０５の孤立セグメント、及び新ＥＸＥファイル１０６の孤立セグメントが、セグメント対応付けデータとしてメモリなどに保持される。

【００６６】上述のようにして、図１２のステップＳ１２０１で、全てのサーチが終了した後、セグメント対応付けデータのそれぞれに対して、以下に示されるステップＳ１２０２～Ｓ１２１６の処理が繰り返し実行される。

【００６７】即ち、まず、ステップＳ１２０２で、１つのセグメント対応付けデータがメモリなどから抽出される。セグメント対応付けデータが存在するならステップＳ１２０３の判定はＹＥＳとなり、続いて、ステップＳ１２０４で、そのセグメント対応付けデータがペアセグメントか否かが判定される。

【００６８】セグメント対応付けデータがペアセグメントであってステップＳ１２０４の判定がＹＥＳの場合には、ステップＳ１２０８、Ｓ１２１０、及びＳ１２１２で、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６の両方又は何れか一方にリロケーション情報が設定されているか否かが判定される。

【００６９】旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６の両方にリロケーション情報が設定されている場合は、ステップＳ１２０８及びＳ１２１０の判定がＮＯとなりステップＳ１２１２の判定がＹＥＳとなって、ステップＳ１２１３が実行される。ステップＳ１２１３では、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき、リロケーションテーブル部領域が抽出される。この領域は、セグメントデータの先頭アドレスから、その先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスまでの領域として抽出される。続いて、セグメントデータ対応付け部１０２は、ステップＳ１２１４で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれについて抽出されたリロケーションテーブル部領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理によって、セグメントデータ対応付け部１０２から渡されたリロケーションテーブル部領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００７０】旧ＥＸＥファイル１０５のみにリロケーション情報が設定されている場合は、ステップＳ１２０８の判定がＹＥＳとなり、ステップＳ１２０９が実行される。ステップＳ１２０９では、旧ＥＸＥファイル１０５

のリソーステーブル部領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【００７１】新ＥＸＥファイル１０６のみにリロケーション情報が設定されている場合は、ステップＳ１２０８の判定がＮＯ、ステップＳ１２１０の判定がＹＥＳとなり、ステップＳ１２１１が実行される。ステップＳ１２１１においては、新ＥＸＥファイル１０６のリソーステーブル部領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【００７２】旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６の何れにもリロケーション情報が設定されていない場合は、上述のリロケーションテーブルに関する処理は実行されない。

【００７３】以上のステップＳ１２１４、Ｓ１２０９、又はＳ１２１１の処理の後、又はステップＳ１２１２の判定がＮＯとなった後、ステップＳ１２１５が実行される。ステップＳ１２１５では、リロケーション情報を除いたセグメントデータ本体の領域が抽出される。この領域は、リロケーション情報が設定されていない場合にはセグメントデータの先頭アドレスから、リロケーション情報が設定されている場合にはセグメントデータの先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスの次のアドレスから、セグメント情報１０８から抽出される次のセグメント番号のセグメントデータの先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される。

【００７４】続いて、セグメントデータ対応付け部１０２は、ステップＳ１２１６で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき抽出されたセグメントデータ領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理によって、セグメントデータ対応付け部１０２から渡されたセグメントデータ領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００７５】一方、前述したステップＳ１２０４の判定において、セグメント対応付けデータがペアセグメントでない場合であってステップＳ１２０４の判定がＮＯの場合には、更にステップＳ１２０５で、セグメント対応付けデータが旧ＥＸＥファイル１０５の孤立セグメントであるか否かが判定される。

【００７６】セグメント対応付けデータが旧ＥＸＥファイル１０５の孤立セグメントであってステップＳ１２０５の判定がＹＥＳなら、ステップＳ１２０６が実行される。ステップＳ１２０６では、旧ＥＸＥファイル１０５の孤立セグメントの領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【００７７】セグメント対応付けデータが旧ＥＸＥファイル１０５の孤立セグメントでなくステップＳ１２０５の判定がＮＯなら、それは新ＥＸＥファイル１０６の孤立セグメントであり、この場合には、ステップＳ１２０７が実行される。ステップＳ１２０７では、新ＥＸＥファイル１０６の孤立セグメントの領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【００７８】以上のステップＳ１２０２～Ｓ１２１６の処理が、全てのセグメント対応付けデータに対して繰り返し実行される。そして、セグメント対応付けデータがなくなると、ステップＳ１２０３の判定がＮＯとなる。このようにして、セグメントデータ対応付け部１０２によって、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との間で、セグメントデータの対応付けが行われた上で、その対応付けされたセグメントデータ領域毎に差分抽出処理部１０４で差分抽出処理が実行される。

【００７９】そして、セグメントデータ対応付け部１０２は、ステップＳ１２１７で、図１のリソース対応付け部１０３を起動して、処理を終了する。リソース対応付け部１０３で実行される処理の動作フローチャートは、図１６に示される。このリソース対応付け部１０３の動作は、ヘッダ解析部１０１などの場合と同様、特には図示しないコンピュータシステムにおいて、ＣＰＵが、ディスク装置などに記憶された旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

【００８０】ここで、リソースはセグメントデータの一属性として規定され、従って、前述したセグメントデータの場合と同様、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６とで対応するリソースがある場合、変更されていないリソースについてはそれらの内容が一致し、変更されたリソースについてのみ差分が生ずる。そこで、リソース対応付け部１０３でも、セグメントデータ対応付け部１０２の場合と同様に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６との間で、リソースの対応付けを行い、その対応付けしたリソース領域毎に差分抽出処理部１０４に対して差分抽出処理を指示する。

【００８１】この場合、特にリソースでは、前述したリソーステーブル部（図１１参照）において、リソースネームＩＤが設定されている。そこで、リソース対応付け部１０３は、リソースネームＩＤを比較することによりその対応付けを行い、その対応付けしたリソース領域毎に差分抽出処理部１０４に対して差分抽出処理を指示する。

【００８２】まず、ステップＳ１６０１で、前述した図４のステップＳ３１０及びＳ３１９でヘッダ解析部１０１から設定されたセグメント情報１０８及びリソース情

報１０９に基づいて、リソースの対応付けが行われる。

【００８３】この処理では、まず、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のセグメント情報１０８（図７参照）及びリソース情報１０９に基づいて、それらのリソースが抽出される。ここで、｛（セグメント情報１０８として設定されているセグメントの数）－（リソース情報１０９として設定されているリソースの数）＋１｝番目以降のセグメント情報１０８が、リソース情報１０９として設定されている各リソースネームＩＤに対応するリソースを示している。旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６において抽出されたリソースの例を図１７(a) に示す。

【００８４】次に、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６において抽出されたリソースが、セグメント情報１０８から抽出されるファイルオフセット順にソートされる。ソートされた旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６のリソースの例を図１７(b) に示す。

【００８５】次に、ソートされた旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６のリソースにおいて、同じリソースネームＩＤを有するもの同士が対応づけられる。この結果、図１７の例における対応関係は、図１８に示される如くとなる。この例では、旧ＥＸＥファイル１０５のリソースネームＩＤ＝６のリソースが削除され、新ＥＸＥファイル１０６のリソースネームＩＤ＝９及び３８のリソースが追加され、その他のリソースのペアについては、変更がないか、又は置換がなされている。

【００８６】そこで、ソートされた旧ＥＸＥファイル１０５のリソース及び新ＥＸＥファイル１０６のリソースについて、以下の対応付け処理がなされる。

処理１：旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のリソース数が比較される。

【００８７】処理２：リソース数が同じか、新ＥＸＥファイル１０６のリソース数の方が多い場合は、旧ＥＸＥファイル１０５で定義されているリソースをキーとして、新ＥＸＥファイル１０６中のリソースがサーチされる。旧ＥＸＥファイル１０５のリソース数の方が多い場合は、新ＥＸＥファイル１０６で定義されているリソースをキーとして、旧ＥＸＥファイル１０５中のリソースがサーチされる。

【００８８】処理３：サーチ対象であるリソースにおいて、現在のキーとされているリソースに対し、リソースネームＩＤが同じものがペアリソースとされる。ここで、前述したセグメントデータの場合と同様、図１５に示されるように、先頭にリロケーション情報が設定されているリロケーションテーブルを有するリソースが存在する。キーとされているリソース又はサーチ対象であるリソースの何れかがこのようなデータである場合には、そのリロケーションテーブルについては、後述するように別に差分抽出処理が実行される。なお、リロケーションテーブルが存在するか否かは、リソースに対応するセグメント情報１０８のセグメントタイプ／属性データに、リロケーションテーブルが存在することを示すデータ値が設定されているか否かによって判別できる。

【００８９】サーチ対象であるリソースにおいて、上記一致関係が成立するリソースがない場合には、キーとなったリソースについては、サーチは諦められ、そのリソースは旧ＥＸＥファイル１０５の孤立リソースとされる。そして、次のリソースがキーとされて、サーチが行われる。

【００９０】全てのサーチが終了した時点で、新ＥＸＥファイル１０６においてペアとならなかったリソースは、新ＥＸＥファイル１０６の孤立リソースとされる。このようにして得られた、ペアリソース、旧ＥＸＥファイル１０５の孤立リソース、及び新ＥＸＥファイル１０６の孤立リソースが、リソース対応付けデータとしてメモリなどに保持される。

【００９１】上述のようにして、図１６のステップＳ１６０１で、全てのサーチが終了した後、リソース対応付けデータのそれぞれに対して、以下に示されるステップＳ１６０２～Ｓ１６１６の処理が繰り返し実行される。これらの処理波、セグメント対応付けデータに対する図１２のステップＳ１２０２～Ｓ１２１６の処理と同じである。

【００９２】即ち、まず、ステップＳ１６０２で、１つのリソース対応付けデータがメモリなどから抽出される。リソース対応付けデータが存在するならステップＳ１６０３の判定はＹＥＳとなり、続いて、ステップＳ１６０４で、そのリソース対応付けデータがペアリソースか否かが判定される。

【００９３】リソース対応付けデータがペアリソースであってステップＳ１６０４の判定がＹＥＳの場合には、ステップＳ１６０８、Ｓ１６１０、及びＳ１６１２で、旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６の両方又は何れか一方にリロケーション情報が設定されているか否かが判定される。

【００９４】旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６の両方にリロケーション情報が設定されている場合は、ステップＳ１６０８及びＳ１６１０の判定がＮＯとなりステップＳ１６１２の判定がＹＥＳとなって、ステップＳ１６１３が実行される。ステップＳ１６１３では、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき、リロケーションテーブル部領域が抽出される。この領域は、リソースの先頭アドレスから、その先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスまでの領域として抽出される。続いて、リソース対応付け部１０３は、ステップＳ１６１４で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６

のそれぞれについて抽出されたリロケーションテーブル部領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理によって、リソース対応付け部１０３から渡されたリロケーションテーブル部領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【００９５】旧ＥＸＥファイル１０５のみにリロケーション情報が設定されている場合は、ステップＳ１６０８の判定がＹＥＳとなり、ステップＳ１６０９が実行される。ステップＳ１６０９では、旧ＥＸＥファイル１０５のリソーステーブル部領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【００９６】新ＥＸＥファイル１０６のみにリロケーション情報が設定されている場合は、ステップＳ１６０８の判定がＮＯ、ステップＳ１６１０の判定がＹＥＳとなり、ステップＳ１６１１が実行される。ステップＳ１６１１においては、新ＥＸＥファイル１０６のリソーステーブル部領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【００９７】旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６の何れにもリロケーション情報が設定されていない場合は、上述のリロケーションテーブルに関する処理は実行されない。

【００９８】以上のステップＳ１６１４、Ｓ１６０９、又はＳ１６１１の処理の後、又はステップＳ１６１２の判定がＮＯとなった後、ステップＳ１６１５が実行される。ステップＳ１６１５では、リロケーション情報を除いたリソース本体の領域が抽出される。この領域は、リロケーション情報が設定されていない場合にはリソースの先頭アドレスから、リロケーション情報が設定されている場合にはリソースの先頭１バイトに設定されているリロケーション項目数分だけ進んだアドレスの次のアドレスから、セグメント情報１０８及びリソース情報１０８から抽出される次のリソース番号のリソースの先頭のオフセットアドレスの手前のアドレスまでの領域として抽出される。

【００９９】続いて、リソース対応付け部１０３は、ステップＳ１６１６で、差分抽出処理部１０４をコールすると共に、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のそれぞれにつき抽出されたリソース領域を差分抽出処理部１０４に渡す。この結果、差分抽出処理部１０４は、後述する差分抽出処理によって、リソース対応付け部１０３から渡されたリソース領域の先頭から末尾まで、新ＥＸＥファイル１０６の旧ＥＸＥファイル１０５に対するバイト単位の差分を抽出し、その差分を更新ファイル１０７にセーブする。

【０１００】一方、前述したステップＳ１６０４の判定において、リソース対応付けデータがペアリソースでない場合であってステップＳ１６０４の判定がＮＯの場合には、更にステップＳ１６０５で、リソース対応付けデータが旧ＥＸＥファイル１０５の孤立リソースであるか否かが判定される。

【０１０１】リソース対応付けデータが旧ＥＸＥファイル１０５の孤立リソースであってステップＳ１６０５の判定がＹＥＳなら、ステップＳ１６０６が実行される。ステップＳ１６０６では、旧ＥＸＥファイル１０５の孤立リソースの領域が削除された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【０１０２】リソース対応付けデータが旧ＥＸＥファイル１０５の孤立リソースでなくステップＳ１６０５の判定がＮＯなら、それは新ＥＸＥファイル１０６の孤立リソースであり、この場合には、ステップＳ１６０７が実行される。ステップＳ１６０７では、新ＥＸＥファイル１０６の孤立リソースの領域が挿入された旨を示す差分情報（更新レコード）が、更新ファイル１０７にセーブされる。

【０１０３】以上のステップＳ１６０２～Ｓ１６１６の処理が、全てのリソース対応付けデータに対して繰り返し実行される。そして、リソース対応付けデータがなくなると、ステップＳ１６０３の判定がＮＯとなり、全ての処理を終了する。

【０１０４】最後に、上述したようにしてヘッダ解析部１０１、セグメントデータ対応付け部１０２、又はリソース対応付け部１０３からコールされる差分抽出処理部１０４の処理について説明する。

【０１０５】差分抽出処理部１０４で実行される差分抽出処理の動作フローチャートは図１９に、また、差分抽出処理において実行される比較処理の動作フローチャートは図２０に示される。これらの動作は、ヘッダ解析部１０１などの場合と同様、特には図示しないコンピュータシステムにおいて、ＣＰＵが、ディスク装置などに記憶された旧ＥＸＥファイル１０５及び新ＥＸＥファイル１０６に対して、メモリに記憶された制御プログラムを実行する動作の一部として実現される。

【０１０６】今、例えば図２２(a) に示されるように、旧ＥＸＥファイル１０５の領域として、バイトデータ列ａｂ・・・ｃｄｅ・・・・からなる領域が指定され、一方、新ＥＸＥファイル１０６の領域として、バイトデータｃとｄの間にバイトデータ列ＩＩ・・・Ｉが挿入された領域が指定されたとする。

【０１０７】図１９において、まず、ステップＳ１９０１で、指定された領域において比較すべきデータがなくなったか否かが判定される。比較すべきデータがある場合、ステップＳ１９０２で、新ＥＸＥファイル１０６において指定された領域を比較側、旧ＥＸＥファイル１０５において指定された領域を比較対象側として、図２１に示されるように、比較側のバイトデータをキーとして

比較処理が実行される。

【０１０８】比較処理の動作フローチャートは図２０に示される。図２０のステップＳ２００１で、特には図示しないメモリに確保されている変数srcendに、比較側の領域における終了位置又は後述する現在の比較側の比較開始位置より１０２４バイト前方の位置のうち、現在の比較側の比較開始位置から近い方の位置のアドレスが格納される。同様にして、特には図示しないメモリに確保されている変数distend に、比較対象側の領域における終了位置又は後述する現在の比較対象側の比較開始位置より１０２４バイト前方の位置のうち、現在の比較対象側の比較開始位置から近い方の位置のアドレスが格納される。

【０１０９】次に、ステップＳ２００２で、特には図示しないメモリに確保されている変数変数src に、現在の比較側の比較開始位置のアドレスが格納される。図２２(a)の例では、変数src には最初は新ＥＸＥファイル１０６の指定領域における先頭のバイトデータａのアドレスが格納される。同様にして、特には図示しないメモリに確保される変数distとnextdistに、現在の比較対象側の比較開始位置のアドレスが格納される。図２２(a) の例では、変数distとnextdistには最初は旧ＥＸＥファイル１０５の指定領域における先頭のバイトデータａのアドレスが格納される。

【０１１０】その後、図２２(a) に示されるように、新ＥＸＥファイル１０６と旧ＥＸＥファイル１０５とで内容が一致しているバイトデータａｂ・・・ｃの範囲では、ステップＳ２００６で変数src 、dist、及びnextdistの値が順次インクリメントされながら、ステップＳ２００３→Ｓ２００４→Ｓ２００５→Ｓ２００６→Ｓ２００３の処理が繰り返されることにより、図２２(a) のＴ１～Ｔ３に示されるように、変数src 及びdistで示されるアドレスにおける値が一致するバイトデータ同士が順次対応付けられてゆく。

【０１１１】そして、図２２(a) に示されるように、ステップＳ２００６で、比較側である新ＥＸＥファイル１０６において変数src の値が挿入されたバイトデータＩＩ・・・Ｉのうち最初のデータのアドレスにされ、比較対象側である旧ＥＸＥファイル１０５において変数distの値がバイトデータｄのアドレスにされた後、次にステップＳ２００４が実行される時点で、その判定がＮＯとなる。

【０１１２】この後は、ステップＳ２００８で変数distの値がインクリメントされながら、ステップＳ２００４→Ｓ２００７→Ｓ２００８→Ｓ２００９→Ｓ２００４→Ｓ２００７→Ｓ２００８の処理が繰り返されることにより、変数src で示されるアドレスにおける新ＥＸＥファイル１０６で挿入された第１番目のバイトデータＩをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける旧ＥＸＥファイル１０５上のバイトデータｄｅ・・・の各値とが、それらが一致してステップＳ２００４の判定がＹＥＳとなるまで、順次比較される。この動作は、図２２(a) のＴ４、Ｔ５に示される。

【０１１３】この比較処理の繰り返しの結果、新ＥＸＥファイル１０６で挿入された第１番目のバイトデータＩに一致するバイトデータが旧ＥＸＥファイル１０５上で見つかれなければ、やがて変数distで示されるアドレスが変数distend で示される比較終了アドレスを越え、ステップＳ２００９の判定がＹＥＳとなる。

【０１１４】この場合には、新ＥＸＥファイル１０６で挿入された第１番目のバイトデータＩをキーとするサーチは諦められ、ステップＳ２０１０で、変数src の値がインクリメントされることにより新ＥＸＥファイル１０６で挿入された第２番目のバイトデータＩがキーとされ、また、変数distに変数nextdistのアドレスが挿入されることにより、比較対象側である旧ＥＸＥファイル１０５上の比較開始位置が初めて不一致が発生したバイトデータｄのアドレスに戻される。

【０１１５】そして、変数src で示されるアドレスが変数srcendで示される比較終了アドレスを越えておらずステップＳ２００３の判定がＮＯであれば、再び、ステップＳ２００８で変数distの値がインクリメントされながら、ステップＳ２００４→Ｓ２００７→Ｓ２００８→Ｓ２００９→Ｓ２００４→Ｓ２００７→Ｓ２００８の処理が繰り返されることにより、変数src で示されるアドレスにおける新ＥＸＥファイル１０６で挿入された第２番目のバイトデータＩをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける旧ＥＸＥファイル１０５上のバイトデータｄｅ・・・の各値とが、それらが一致してステップＳ２００４の判定がＹＥＳとなるまで、順次比較される。この動作は、図２２(a)のＴ６、Ｔ７に示される。

【０１１６】以下、ステップＳ２００９の判定がＹＥＳとなる毎に、新ＥＸＥファイル１０６で挿入された第３番目以降の各バイトデータＩについて同様の比較処理が繰り返される（図２２(a) のＴ８、Ｔ９）。

【０１１７】やがてステップＳ２０１０で、変数src に新ＥＸＥファイル１０６上のバイトデータｄのアドレスが代入され、変数distに変数nextdistで示される旧ＥＸＥファイル１０５上のバイトデータｄのアドレスが代入された後、ステップＳ２００４が実行された時点で、変数src で示されるアドレスにおける新ＥＸＥファイル１０６のバイトデータｄと変数distで示されるアドレスにおける旧ＥＸＥファイル１０５のバイトデータｄとが一致し、その判定がＹＥＳとなる。この一致状態は、最初に新ＥＸＥファイル１０６と旧ＥＸＥファイル１０５のバイトデータａｂ・・・ｃが一致し、次に、新ＥＸＥファイル１０６で挿入されたバイトデータＩＩ・・・Ｉが旧ＥＸＥファイル１０５上のバイトデータｄｅ・・・と

不一致となった後の一致状態であるため、ステップＳ２００５の判定がＹＥＳとなる。

【０１１８】この結果、ステップＳ２０１１で、図２２(a) の新ＥＸＥファイル１０６と旧ＥＸＥファイル１０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点（バイトデータａのアドレス）と一致バイト数が算出され、また、新ＥＸＥファイル１０６で挿入されたバイトデータＩＩ・・・Ｉで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１１９】上述のように、一致部分と不一致部分のペアが見つかってステップＳ２００５の判定がＹＥＳとなる時点で、また、図２６の説明で後述するように、不一致部分と一致部分のペアが見つかってステップＳ２００７の判定がＹＥＳとなる時点で、一致部分の一致開始点と一致バイト数、及び不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了される。

【０１２０】以上の動作の結果、新ＥＸＥファイル１０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル１０５の指定領域の各バイトデータに対する図１９のステップＳ１９０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

【０１２１】一致バイト数が６４バイト以上であってステップＳ１９０３の判定がＹＥＳとなった場合には、ステップＳ１９０９で、上述の不一致部分に関する差分情報が図１の更新ファイル１０７へライトされる。例えば図２２(a) の新ＥＸＥファイル１０６と旧ＥＸＥファイル１０５のバイトデータａｂ・・・ｃで示される一致部分の一致バイト数が６４バイト以上である場合には、新ＥＸＥファイル１０６において、旧ＥＸＥファイル１０５のバイトデータｃで示されるアドレスの直後にバイトデータＩＩ・・・Ｉが挿入されたことを示す差分情報が更新ファイル１０７へライトされる。

【０１２２】一致バイト数が６４バイトより少なくステップＳ１９０３の判定がＮＯとなった場合には、ステップＳ１９０４で、前述したステップＳ１９０２の場合とは逆に、旧ＥＸＥファイル１０５の指定領域を比較側、新ＥＸＥファイル１０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述した図２０で示される比較処理が実行される。

【０１２３】この比較処理の例として、前述した図２２(a) に対応する動作が図２２(b) に示される。まず、ステップＳ２００１とステップＳ２００２で、変数srcendとdistend 、変数src 、及び変数distとnextdistに、図２２(a) の場合と同様の値が設定される。

【０１２４】その後、前述したステップＳ２００３→Ｓ２００４→Ｓ２００５→Ｓ２００６→Ｓ２００３の処理が繰り返されることにより、図２２(b) のＴ１～Ｔ３に示されるように、変数src 及びdistで示されるアドレスにおける値が一致するバイトデータ同士が順次対応付けられてゆく。

【０１２５】そして、図２２(b) に示されるように、ステップＳ２００６で、比較側である旧ＥＸＥファイル１０５において変数src の値がｄのアドレスにされ、比較対象側である新ＥＸＥファイル１０６において変数distの値が挿入されたバイトデータＩＩ・・・Ｉのうち最初のデータのアドレスにされた後、次にステップＳ２００４が実行される時点で、その判定がＮＯとなる。

【０１２６】この後は、ステップＳ２００８で変数distの値がインクリメントされながら、ステップＳ２００４→Ｓ２００７→Ｓ２００８→Ｓ２００９→Ｓ２００４→Ｓ２００７→Ｓ２００８の処理が繰り返されることによって、変数src で示されるアドレスにおける旧ＥＸＥファイル１０５のバイトデータｄをキーとして、その値と、順次インクリメントされる変数distで示されるアドレスにおける新ＥＸＥファイル１０６上のバイトデータＩＩ・・・Ｉの各値とが、それらが一致してステップＳ２００４の判定がＹＥＳとなるまで、順次比較される。この動作は、図２２(b) のＴ４～Ｔ６に示される。

【０１２７】この比較処理の繰り返しの結果、ステップＳ２００８で、変数distに新ＥＸＥファイル１０６上のバイトデータｄのアドレスが代入された後、ステップＳ２００４が実行された時点で、変数src で示されるアドレスにおける旧ＥＸＥファイル１０５のバイトデータｄと変数distで示されるアドレスにおける新ＥＸＥファイル１０６のバイトデータｄとが一致し、その判定がＹＥＳとなる。この一致状態は、最初に旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のバイトデータａｂ・・・ｃが一致し、次に、新ＥＸＥファイル１０６で挿入されたバイトデータＩＩ・・・Ｉが旧ＥＸＥファイル１０５上のバイトデータｄと不一致となった後の一致状態であるため、ステップＳ２００５の判定がＹＥＳとなる。

【０１２８】この結果、ステップＳ２０１１で、図２２(b) の旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点（バイトデータａのアドレス）と一致バイト数が算出され、また、新ＥＸＥファイル１０６で挿入されたバイトデータＩＩ・・・Ｉで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１２９】以上の動作の結果、旧ＥＸＥファイル１０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル１０６の指定領域の各バイトデータに対する図１９のステップＳ１９０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０５で、全てのバイトが不一致であるか否かが判定される。

【０１３０】全てのバイトが不一致であってステップＳ

１９０５の判定がＹＥＳとなった場合には、ステップＳ１９０９で、上述の不一致部分に関する差分情報が図１の更新ファイル１０７へライトされる。

【０１３１】一方、図２２の例のように、全てのバイトが不一致でなくステップＳ１９０５の判定がＮＯとなった場合には、ステップＳ１９０６で、まず、次式によって、新ＥＸＥファイル１０６のバイトデータをキーとする前述したステップＳ１９０２の比較処理における一致バイト数の割合RNEWOLD が算出される。

【０１３２】

【数１】RNEWOLD=（新→旧の一致バイト数）／｛（新→旧の一致バイト数＋（新→旧の不一致バイト数）｝

次に、次式によって、旧ＥＸＥファイル１０５のバイトデータをキーとする前述したステップＳ１９０４の比較処理における一致バイト数の割合ROLDNEW が算出される。

【０１３３】

【数２】ROLDNEW=（旧→新の一致バイト数）／｛（旧→新の一致バイト数＋（旧→新の不一致バイト数）｝

そして、次式によって、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

【０１３４】

【数３】RNEWOLD/ROLDNEW ＜１　？

今、更新ファイル１０７には、不一致部分の差分情報が格納されるため、一致部分はなるべく多い方がよい。従って、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW より小さくステップＳ１９０６の判定がＹＥＳの場合には、ステップＳ１９０７で、旧ＥＸＥファイル１０５をキーとした場合のステップＳ１９０４の比較処理の結果が選択され、ステップＳ１９０９で、その比較処理によって得られている不一致部分に関する差分情報が図１の更新ファイル１０７へライトされる。一方、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW 以上であってステップＳ１９０６の判定がＮＯの場合には、ステップＳ１９０８で、新ＥＸＥファイル１０６をキーとした場合のステップＳ１９０２の比較処理の結果が選択され、ステップＳ１９０９で、その比較処理によって得られている不一致部分に関する差分情報が図１の更新ファイル１０７へライトされる。

【０１３５】但し、図２２(a) と(b) の例では、RNEWOLD=ROLDNEW となるため、ステップＳ１９０８で、新ＥＸＥファイル１０６をキーとした場合のステップＳ１９０２の比較処理の結果が選択され、ステップＳ１９０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル１０６において旧ＥＸＥファイル１０５のバイトデータｃで示されるアドレスの直後にバイトデータＩＩ・・・Ｉが挿入されたことを示す差分情報が、更新ファイル１０７へライトされる。

【０１３６】ステップＳ１９０９の処理の後は、ステップＳ１９１０で、次の比較開始位置が設定され、ステップＳ１９０１以降の処理に戻る。図２２(a) の例では、次の比較開始位置は、新ＥＸＥファイル１０６では、最後に図２０のステップＳ２００４の判定がＹＥＳとなったときの変数src に格納されているバイトデータｄのアドレス、旧ＥＸＥファイル１０５でも、同じくそのときの変数distに格納されているバイトデータｄのアドレスである。以後、これらの比較開始位置を起点として、前述した場合と全く同様の比較処理が繰り返される。

【０１３７】新ＥＸＥファイル１０６の指定領域と旧ＥＸＥファイル１０５の指定領域において、比較するバイトデータが存在しなくなったら、ステップＳ１９０１の判定がＹＥＳとなって、これらの指定領域における差分抽出処理を終了する。

【０１３８】次に、図２３(a) に示されるように、新ＥＸＥファイル１０６の指定領域において、旧ＥＸＥファイル１０５の指定領域のバイトデータｃとｄの間に存在したバイトデータ列ＤＤ・・・Ｄの領域が削除された場合を考える。

【０１３９】この場合、まず、図２３(a) に示されるように、図１９のステップＳ１９０２で、新ＥＸＥファイル１０６において指定された領域を比較側、旧ＥＸＥファイル１０５において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。

【０１４０】この比較処理は、前述した図２２(b) の場合に対し、比較側と比較対象側が逆になっただけで、同様の処理である。そして、この比較処理の結果、図２３(a)の新ＥＸＥファイル１０６と旧ＥＸＥファイル１０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル１０６で削除されたバイトデータＤＤ・・・Ｄで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１４１】以上の動作の結果、新ＥＸＥファイル１０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル１０５の指定領域の各バイトデータに対する図１９のステップＳ１９０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

【０１４２】一致バイト数が６４バイト以上であってステップＳ１９０３の判定がＹＥＳとなった場合には、ステップＳ１９０９で、上述の不一致部分に関する差分情

報が図１の更新ファイル１０７へライトされる。

【０１４３】一致バイト数が６４バイトより少なくステップＳ１９０３の判定がＮＯとなった場合には、ステップＳ１９０４で、前述したステップＳ１９０２の場合とは逆に、旧ＥＸＥファイル１０５の指定領域を比較側、新ＥＸＥファイル１０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図２０で示される比較処理が実行される。この比較処理の例として、前述の図２３(a) に対応する動作が図２３(b) に示される。この比較処理は、前述の図２２(a) の場合に対して、比較側と比較対象側が逆になっただけで、同様の処理である。そして、この比較処理の結果、図２３(b) の旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル１０６で削除されたバイトデータＤＤ・・・Ｄで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１４４】以上の動作の結果、旧ＥＸＥファイル１０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル１０６の指定領域の各バイトデータに対する図１９のステップＳ１９０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０５で、全てのバイトが不一致であるか否かが判定される。

【０１４５】図２３の例では、全てのバイトが不一致でなくステップＳ１９０５の判定がＮＯとなるため、ステップＳ１９０６で、前述した数1式、数2式、及び数3式に基づいて、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

【０１４６】図２３(a) と(b) の例では、RNEWOLD=ROLDNEW となるため、ステップＳ１９０８で、新ＥＸＥファイル１０６をキーとした場合のステップＳ１９０２の比較処理の結果が選択され、ステップＳ１９０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル１０６において旧ＥＸＥファイル１０５のバイトデータｃとｄの間のバイトデータＤＤ・・・Ｄが削除されたことを示す差分情報が、更新ファイル１０７へライトされる。

【０１４７】次に、図２４(a) に示されるように、新ＥＸＥファイル１０６の指定領域において、旧ＥＸＥファイル１０５の指定領域のバイトデータｃとｄの間に存在したバイトデータ列ＲＲ・・・Ｒの領域がそれと同じ長さを有する他のバイトデータ列ＲＲ・・・Ｒに置換された場合を考える。

【０１４８】この場合、まず、図２４(a) に示されるように、図１９のステップＳ１９０２で、新ＥＸＥファイル１０６において指定された領域を比較側、旧ＥＸＥファイル１０５において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、まず、図２４(a) のＴ１～Ｔ９で示される比較範囲では、前述の図２２(a) のＴ１～Ｔ９で示される比較範囲のものと同様の処理であり、続いて、図２４(a) のＴ１０～Ｔ１３で示される比較範囲では、前述の図２３(a) のＴ４～Ｔ７で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図２４(a) の新ＥＸＥファイル１０６と旧ＥＸＥファイル１０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル１０６で置換されたバイトデータＲＲ・・・Ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１４９】以上の動作の結果、新ＥＸＥファイル１０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル１０５の指定領域の各バイトデータに対する図１９のステップＳ１９０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

【０１５０】一致バイト数が６４バイト以上であってステップＳ１９０３の判定がＹＥＳとなった場合には、ステップＳ１９０９で、上述の不一致部分に関する差分情報が図１の更新ファイル１０７へライトされる。

【０１５１】一致バイト数が６４バイトより少なくステップＳ１９０３の判定がＮＯとなった場合には、ステップＳ１９０４で、前述したステップＳ１９０２の場合とは逆に、旧ＥＸＥファイル１０５の指定領域を比較側、新ＥＸＥファイル１０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図２０で示される比較処理が実行される。この比較処理の例として、前述の図２４(a) に対応する動作が図２４(b) に示される。この比較処理は、まず、図２４(b) のＴ１～Ｔ９で示される比較範囲では、前述した図２３(b) のＴ１～Ｔ９で示される比較範囲のものと同様の処理であって、続いて、図２４(b) のＴ１０～Ｔ１３で示される比較範囲では、前述した図２２(b) のＴ４～Ｔ７で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図２４(b) の旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル１０６で置換されたバイトデータＲＲ・・・Ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１５２】以上の動作の結果、旧ＥＸＥファイル１０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル１０６の指定領域の各バイトデータに対する図１９のステップＳ１９０４の比較処理によって、一致部分

と不一致部分のペアが見つかると、次にステップＳ１９０５で、全てのバイトが不一致であるか否かが判定される。

【０１５３】図２４の例では、全てのバイトが不一致でなくステップＳ１９０５の判定がＮＯとなるため、ステップＳ１９０６で、前述した数１式、数２式、及び数３式に基づいて、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

【０１５４】図２４(a) と(b) の例では、RNEWOLD=ROLDNEW となるため、ステップＳ１９０８で、新ＥＸＥファイル１０６をキーとした場合のステップＳ１９０２の比較処理の結果が選択され、ステップＳ１９０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル１０６において旧ＥＸＥファイル１０５のバイトデータｃとｄの間のバイトデータＲＲ・・・Ｒが置換されたことを示す差分情報が、更新ファイル１０７へライトされる。

【０１５５】次に、図２５(a) に示されるように、新ＥＸＥファイル１０６の指定領域において、旧ＥＸＥファイル１０５の指定領域のバイトデータｃとｄの間に存在するバイトデータ列のうち、バイトデータ列Ｒ・・・Ｒ及びｒ・・・ｒの領域がそれと同じ長さを有する他のバイトデータ列Ｒ・・・Ｒ及びｒ・・・ｒに置換され、バイトデータ列ＤＤ・・・Ｄが削除され、また、旧ＥＸＥファイル１０５の指定領域のバイトデータｃとｄの間に、そこに存在していなかった新たなバイトデータ列ＩＩ・・・Ｉが挿入された場合を考える。

【０１５６】この場合、まず、図２５(a) に示されるように、図１９のステップＳ１９０２で、新ＥＸＥファイル１０６において指定された領域を比較側、旧ＥＸＥファイル１０５において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、図２４(a) の場合と同様であり、従って、図２５(a) のＴ１～Ｔ１０で示される比較範囲では、前述の図２２(a) のＴ１～Ｔ９で示される比較範囲のものと同様の処理であり、続いて、図２５(a) のＴ１１～Ｔ１３で示される比較範囲では、前述の図２３(a) のＴ４～Ｔ７で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図２５(a) の新ＥＸＥファイル１０６と旧ＥＸＥファイル１０５のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル１０６で挿入、削除、及び置換されたバイトデータＩ・・・ＩＲ・・・Ｒｒ・・・ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１５７】以上の動作の結果、新ＥＸＥファイル１０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル１０５の指定領域の各バイトデータに対する図１９のステップＳ１９０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

【０１５８】一致バイト数が６４バイト以上であってステップＳ１９０３の判定がＹＥＳとなった場合には、ステップＳ１９０９で、上述の不一致部分に関する差分情報が図１の更新ファイル１０７へライトされる。

【０１５９】一致バイト数が６４バイトより少なくステップＳ１９０３の判定がＮＯとなった場合には、ステップＳ１９０４で、前述したステップＳ１９０２の場合とは逆に、旧ＥＸＥファイル１０５の指定領域を比較側、新ＥＸＥファイル１０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図２０で示される比較処理が実行される。この比較処理の例として、前述の図２５(a) に対応する動作が図２５(b) に示される。この比較処理は、前述の図２４(b) の場合と同様であり、まず、図２５(b) のＴ１～Ｔ９で示される比較範囲では、前述した図２３(b) のＴ１～Ｔ９で示される比較範囲のものと同様の処理であって、続いて、図２５(b) のＴ１０～Ｔ１２で示される比較範囲では、前述した図２２(b) のＴ４～Ｔ７で示される比較範囲のものと同様の処理である。そして、この比較処理の結果、図２５(b) の旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出され、また、新ＥＸＥファイル１０６で挿入、削除、及び置換されたバイトデータＩ・・・ＩＲ・・・Ｒｒ・・・ｒで示される不一致部分の不一致開始点と不一致バイト数が算出されて、比較処理が終了する。

【０１６０】以上の動作の結果、旧ＥＸＥファイル１０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル１０６の指定領域の各バイトデータに対する図１９のステップＳ１９０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０５で、全てのバイトが不一致であるか否かが判定される。

【０１６１】図２５の例では、全てのバイトが不一致でなくステップＳ１９０５の判定がＮＯとなるため、ステップＳ１９０６で、前述した数１式、数２式、及び数３式に基づいて、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

【０１６２】図２５(a) と(b) の例では、RNEWOLD=ROLDNEW となるため、ステップＳ１９０８で、新ＥＸＥファイル１０６をキーとした場合のステップＳ１９０２の比較処理の結果が選択され、ステップＳ１９０９で、その比較処理によって得られている不一致部分に関する差分

情報が更新ファイル１０７へライトされる。

【０１６３】ここで、図２５(a) において、旧ＥＸＥファイル１０５の指定領域におけるバイトデータｃとｄの間のバイト数の方が、新ＥＸＥファイル１０６の指定領域におけるバイトデータｃとｄの間のバイト数より多ければ、まず、新ＥＸＥファイル１０６における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル１０７へライトされ、続いて、｛（旧ＥＸＥファイル１０５における当該バイト数）－（新ＥＸＥファイル１０６における当該バイト数）｝分のバイトデータが削除された旨の差分情報が更新ファイル１０７へライトされる。

【０１６４】逆に、図２５(a) において、旧ＥＸＥファイル１０５の指定領域におけるバイトデータｃとｄの間のバイト数の方が、新ＥＸＥファイル１０６の指定領域におけるバイトデータｃとｄの間のバイト数より少なければ、まず、旧ＥＸＥファイル１０５における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル１０７へライトされ、続いて、｛（新ＥＸＥファイル１０６における当該バイト数）－（旧ＥＸＥファイル１０５における当該バイト数）｝分のバイトデータが挿入された旨の差分情報が更新ファイル１０７へライトされる。

【０１６５】また、図２５(a) において、旧ＥＸＥファイル１０５の指定領域におけるバイトデータｃとｄの間のバイト数が、新ＥＸＥファイル１０６の指定領域におけるバイトデータｃとｄの間のバイト数と等しければ、旧ＥＸＥファイル１０５又は新ＥＸＥファイル１０６における当該バイト数分のバイトデータが置換された旨の差分情報が更新ファイル１０７へライトされる。

【０１６６】最後に、図２６に示されるような特殊な例を考える。この例では、新ＥＸＥファイル１０６の指定領域において、旧ＥＸＥファイル１０５の指定領域の先頭のバイトデータａの手前にバイトデータ列ＩＩ・・・Ｉが挿入され、旧ＥＸＥファイル１０５の指定領域のバイトデータｃとｄの間に存在したバイトデータ列ＤＤ・・・Ｄが削除された場合であって、挿入されたバイトデータ列ＩＩ・・・Ｉと削除されたバイトデータ列ＤＤ・・・Ｄが偶然に一致しているような場合が示されている。

【０１６７】この場合、まず、図２６(a) に示されるように、図１９のステップＳ１９０２で、新ＥＸＥファイル１０６において指定された領域を比較側、旧ＥＸＥファイル１０５において指定された領域を比較対象側として、比較側のバイトデータをキーとして比較処理が実行される。この比較処理は、図２６(a) のＴ１～Ｔ３で示される比較範囲では、前述の図２３(a) のＴ４～Ｔ６で示される比較範囲のものと同様の処理であり、続いて、図２６(a) のＴ４～Ｔ６で示される比較範囲では、前述の図２２(a) のＴ１～Ｔ３で示される比較範囲のものと同様の処理である。なお、図２６(a) のＴ３→Ｔ４の変化時には、最初に新ＥＸＥファイル１０６上の挿入された第１番目のバイトデータＩと旧ＥＸＥファイル１０５上の各バイトデータａｂ・・・ｃとが不一致となった後に、新ＥＸＥファイル１０６上の挿入された第１番目のバイトデータＩと旧ＥＸＥファイル１０５上の新ＥＸＥファイル１０６において削除された第１番目のバイトデータＤとが一致した状態となるため、ステップＳ２００５の判定はＮＯとなる。

【０１６８】そして、図２６(a) のＴ６で示される処理の後、ステップＳ２００６で、変数src に新ＥＸＥファイル１０６上のバイトデータａのアドレスが代入され、変数distに旧ＥＸＥファイル１０５上のバイトデータｄのアドレスが代入されて、ステップＳ２００４が実行された時点で、変数src で示されるアドレスにおける新ＥＸＥファイル１０６のバイトデータａと変数distで示されるアドレスにおける旧ＥＸＥファイル１０５のバイトデータｄとが不一致となり、その判定がＮＯとなる。この不一致状態は、最初に新ＥＸＥファイル１０６上の挿入された各バイトデータＩＩ・・・Ｉと旧ＥＸＥファイル１０５上の各バイトデータａｂ・・・ｃとが不一致となり、更に、新ＥＸＥファイル１０６上の挿入された各バイトデータＩＩ・・・Ｉと旧ＥＸＥファイル１０５上の新ＥＸＥファイル１０６において削除された各バイトデータＤＤ・・・Ｄとが一致した後の不一致状態であるため、ステップＳ２００７の判定はＹＥＳとなって、ステップＳ２０１１が実行される。ステップＳ２０１１では、図２６(a) の旧ＥＸＥファイル１０５上のバイトデータａｂ・・・ｃで示される不一致部分の不一致開始点と不一致バイト数が算出され、また、新ＥＸＥファイル１０６上の挿入されたバイトデータＩＩ・・・Ｉと旧ＥＸＥファイル１０５上の新ＥＸＥファイル１０６で削除されたバイトデータＤＤ・・・Ｄとで示される一致部分の一致開始点と一致バイト数が算出されて、比較処理が終了する。

【０１６９】以上の動作の結果、新ＥＸＥファイル１０６の指定領域の各バイトデータをキーとする旧ＥＸＥファイル１０５の指定領域の各バイトデータに対する図１９のステップＳ１９０２の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０３で、上述の一致バイト数が６４バイト以上であるか否かが判定される。

【０１７０】今、図２６(a) において、一致部分であるバイトデータ列ＩＩ・・・Ｉ（＝ＤＤ・・・Ｄ）のバイト数が例えば数バイト程度と少なく、不一致部分であるバイトデータ列ａｂ・・・ｃのバイト数が多い場合には、一致部分が短く不一致部分が長い結果となる。

【０１７１】このような場合には、ステップＳ１９０３の判定がＮＯとなる結果、ステップＳ１９０４で、前述したステップＳ１９０２の場合とは逆に、旧ＥＸＥファ

イル１０５の指定領域を比較側、新ＥＸＥファイル１０６の指定領域を比較対象側とし、比較側のバイトデータをキーとして、前述の図２０で示される比較処理が実行される。この比較処理の例として、前述の図２６(a) に対応する動作が図２６(b) に示される。この比較処理は、図２６(b) のＴ１～Ｔ３で示される比較範囲では、前述の図２２(b) のＴ４～Ｔ６で示される比較範囲のものと同様の処理であり、続いて、図２６(b) のＴ４～Ｔ６で示される比較範囲では、前述の図２２(b) のＴ１～Ｔ３で示される比較範囲のものと同様の処理である。

【０１７２】そして、図２６(b) のＴ６で示される処理の後、ステップＳ２００６で、変数src に旧ＥＸＥファイル１０５上のバイトデータＤのアドレスが代入され、変数distに新ＥＸＥファイル１０６上のバイトデータｄのアドレスが代入されて、ステップＳ２００４が実行された時点で、変数src で示されるアドレスにおける旧ＥＸＥファイル１０５のバイトデータＤと変数distで示されるアドレスにおける新ＥＸＥファイル１０６のバイトデータｄとが不一致となり、その判定がＮＯとなる。この不一致状態は、最初に旧ＥＸＥファイル１０５のバイトデータａと新ＥＸＥファイル１０６上の挿入された各バイトデータＩＩ・・・Ｉとが不一致となり、更に、旧ＥＸＥファイル１０５上の各バイトデータａｂ・・・ｃと新ＥＸＥファイル１０６上の各バイトデータａｂ・・・ｃとが一致した後の不一致状態であるため、ステップＳ２００７の判定はＹＥＳとなって、ステップＳ２０１１が実行される。ステップＳ２０１１では、図２６(b) の新ＥＸＥファイル１０６上のバイトデータＩＩ・・・Ｉで示される不一致部分の不一致開始点と不一致バイト数が算出され、また、旧ＥＸＥファイル１０５と新ＥＸＥファイル１０６上のバイトデータａｂ・・・ｃで示される一致部分の一致開始点と一致バイト数が算出されて、比較処理が終了する。

【０１７３】以上の動作の結果、旧ＥＸＥファイル１０５の指定領域の各バイトデータをキーとする新ＥＸＥファイル１０６の指定領域の各バイトデータに対する図１９のステップＳ１９０４の比較処理によって、一致部分と不一致部分のペアが見つかると、次にステップＳ１９０５で、全てのバイトが不一致であるか否かが判定される。

【０１７４】図２６の例では、全てのバイトが不一致でなくステップＳ１９０５の判定がＮＯとなるため、ステップＳ１９０６で、前述した数１式、数２式、及び数３式に基づいて、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW より小さいか否かが判定される。

【０１７５】今、前述したように、図２６(b) において、一致部分であるバイトデータ列ａｂ・・・ｃのバイト数が多く、不一致部分であるバイトデータ列ＩＩ・・・Ｉのバイト数が例えば数バイト程度と少ない場合には、一致部分が長く不一致部分が短い結果となる。

【０１７６】従って、図２６(a) と(b) の例では、RNEWOLD ＜ROLDNEW となる。前述したように、更新ファイル１０７には、不一致部分の差分情報が格納されるため、一致部分はなるべく多い方がよい。従って、図２６の場合のように、新ＥＸＥファイル１０６をキーとした場合の一致バイト数の割合RNEWOLD の方が旧ＥＸＥファイル１０５をキーとした場合の一致バイト数の割合ROLDNEW より小さくステップＳ１９０６の判定がＹＥＳの場合には、ステップＳ１９０７で、旧ＥＸＥファイル１０５をキーとした場合のステップＳ１９０４の比較処理の結果が選択され、ステップＳ１９０９で、その比較処理によって得られている不一致部分に関する差分情報として、新ＥＸＥファイル１０６において旧ＥＸＥファイル１０５のバイトデータａで示されるアドレスの手前にバイトデータＩＩ・・・Ｉが挿入されたことを示す差分情報が、更新ファイル１０７へライトされる。

【０１７７】以上説明したようにして、差分抽出処理部１０４において差分抽出処理が実行される。図２７に、更新ファイル１０７のデータ構造を示す。

【０１７８】更新ファイル１０７の各レコードは、図２７(a) に示されるように、そのレコードの属性を示すデータRdcordIDと、旧ＥＸＥファイル１０５に対するそのレコードの処理位置を示すデータDataOffset、更新される実データの長さを示すデータDataLength及び更新される実データ配列Data n　とから構成される。

【０１７９】RdcordIDの属性は、図２７(b) に示される。まず、値が０のRdcordIDは、そのレコードが置換レコードであることを示し、この場合には、旧ＥＸＥファイル１０５上のデータDataOffsetで示される位置から、データDataLengthで示される長さのデータが、実データ配列Data n　で示されるデータに置換される。

【０１８０】また、値が１のRdcordIDは、そのレコードが挿入レコードであることを示し、この場合には、旧ＥＸＥファイル１０５上のデータDataOffsetで示される位置から、データDataLengthで示される長さの実データ配列Data n　で示されるデータが挿入される。

【０１８１】更に、値が２のRdcordIDは、そのレコードが削除レコードであることを示し、この場合には、旧ＥＸＥファイル１０５上のデータDataOffsetで示される位置から、データDataLengthで示される長さのデータが削除される。なお、この場合には、実データ配列Data n は付加されていない。

【０１８２】このように生成された更新ファイル１０７に基づいて旧ＥＸＥファイル１０５が更新される場合の動作は、更新ファイル１０７から１つ１つレコードが取り出され、各レコードに対して上述の各レコード属性毎の処理が実行されることにより、簡単に実現される。

【０１８３】以上説明した実施例は、本発明をOS/2シス

テムのもとで実行可能なＥＸＥファイルの差分抽出処理に適用したものであって、DOS EXE 形式の領域、OS/2 EXEヘッダ領域、各テーブル部領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行される。しかし、本発明はこれに限られるものではなく、他の論理単位、例えばページデータ単位などで対応付けが行われた上で差分抽出処理が実行されるように構成することもできる。

【０１８４】

【発明の効果】本発明によれば、差分抽出処理が実行される場合、例えばOS/2システムにおける実行ファイルでは、DOS EXE 形式の領域、OS/2 EXEヘッダ領域、各テーブル部領域、セグメントデータ領域、及びリソース領域のそれぞれの論理単位毎に、対応付けが行われた上で差分抽出処理が実行されるため、旧実行ファイルと新実行ファイルとの間で、各領域のデータが一致する確率が高くなり、その結果、生成される更新ファイルのサイズを大幅に削減することが可能となる。

【０１８５】これにより、例えばネットワークを介して更新ファイルが配信される場合などで、新ＥＸＥファイルがそのまま配信される場合に比較して回線の使用効率を大幅に向上させることが可能となる。

【図面の簡単な説明】

【図１】本発明の実施例の構成図である。

【図２】本発明の実施例におけるファイル形式を示した図である。

【図３】ヘッダ解析部の動作フローチャート（その１）である。

【図４】ヘッダ解析部の動作フローチャート（その２）である。

【図５】DOS EXE ヘッダとOS/2 EXEヘッダのデータ構造を示した図である。

【図６】OS/2 EXEヘッダ情報を示した図である。

【図７】セグメントテーブル部のデータ構造を示した図である。

【図８】常駐／非常駐ネームテーブル部のデータ構造を示した図である。

【図９】インポートネームテーブル部のデータ構造を示した図である。

【図１０】モジュール参照テーブル部とインポートネームテーブル部の関係を示した図である。

【図１１】リソーステーブル部のデータ構造を示した図である。

【図１２】セグメントデータ対応付け部の動作フローチャートである。

【図１３】旧ＥＸＥファイル及び新ＥＸＥファイルのセグメントの配列を示した図である。

【図１４】セグメントの対応関係の説明図である。

【図１５】リロケーション情報の説明図である。

【図１６】リソース対応付け部の動作フローチャートである。

【図１７】旧ＥＸＥファイルと新ＥＸＥファイルのリソースの配列を示した図である。

【図１８】リソースの対応関係の説明図である。

【図１９】差分抽出処理の動作フローチャートである。

【図２０】比較処理の動作フローチャートである。

【図２１】比較処理の説明図である。

【図２２】新ＥＸＥファイルに挿入が行われた場合の差分抽出処理の説明図である。

【図２３】新ＥＸＥファイルで削除が行われた場合の差分抽出処理の説明図である。

【図２４】新ＥＸＥファイルで置換が行われた場合の差分抽出処理の説明図である。

【図２５】新ＥＸＥファイルで挿入、削除、及び置換が同時に行われた場合の差分抽出処理の説明図である。

【図２６】新ＥＸＥファイルで特殊な挿入及び削除が同時に行われた場合の差分抽出処理の説明図である。

【図２７】更新ファイルのデータ構造を示した図である。

【図２８】ファイル更新の説明図である。

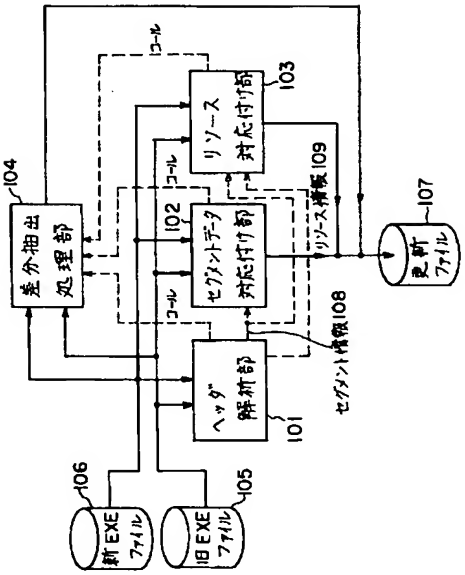
【図２９】ファイル更新環境の説明図である。

【符号の説明】

１０１	ヘッダ解析部
１０２	セグメントデータ対応付け部
１０３	リソース対応付け部
１０４	差分抽出処理部
１０５	旧ＥＸＥファイル
１０６	新ＥＸＥファイル
１０７	更新ファイル
１０８	セグメント情報
１０９	リソース情報

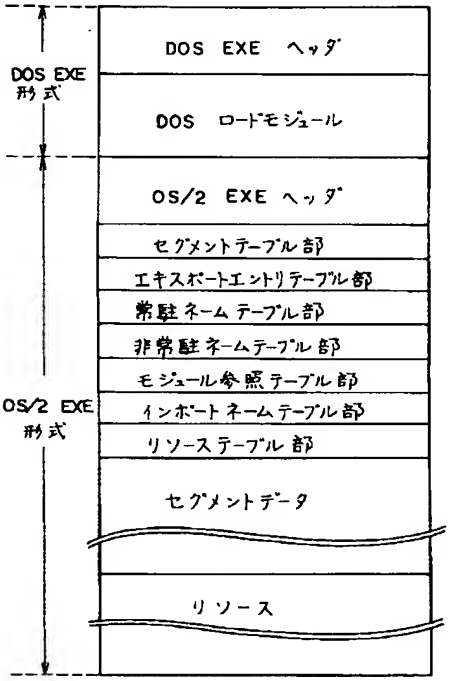
【図1】

本発明の実施例の構成図



【図2】

本発明の実施例におけるファイル形式を示した図



【図8】

常駐/非常駐ネームテーブル部のデータ構造を示した図

サイズ	オフセット	内容
1バイト	0h	文字列のバイト長、0=テーブルエンド
nバイト	1h	名称列
1ワード	n+1h	エントリテーブルのインデックス

【図13】

旧EXEファイル及び新EXEファイルのセグメントの配列を示した図

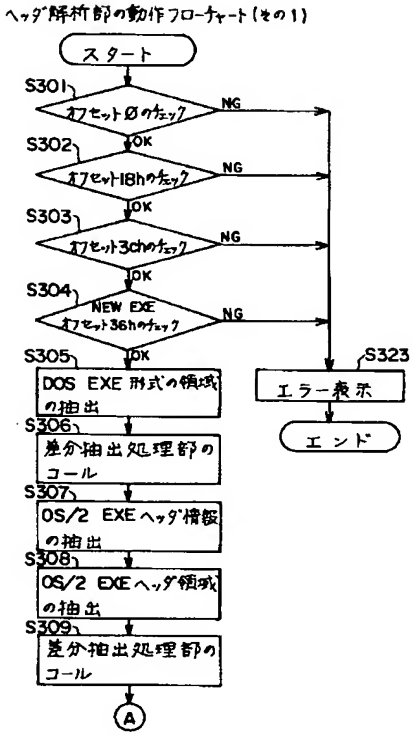
(a) 旧EXEファイル

Seg No.	offset	Size	Type
1	1123	8000	CODE, LOADONCALL...
2	2000	3000	DATA, ...
3	3000	4000	CODE, PRELOAD...
4	3500	3000	DATA, ...

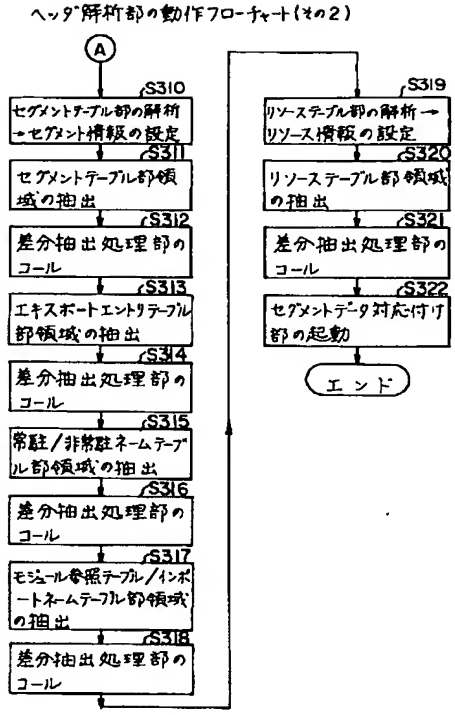
(b) 新EXEファイル

Seg No.	offset	Size	Type
1	1123	8000	CODE, LOADONCALL...
2	2100	3000	DATA, ...
3	3000	4000	DATA, ...
4	4000	4000	CODE, PRELOAD...

【図3】

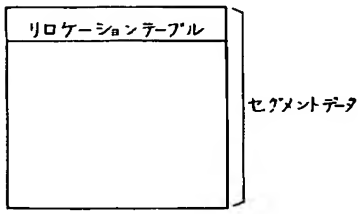


【図4】



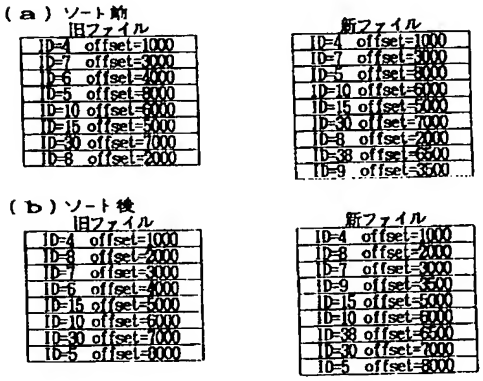
【図15】

リロケーション情報の説明図



【図17】

旧EXEファイルと新EXEファイルのセグメントの配列を示した図



【図５】

DOS EXEヘッダとOS/2EXEヘッダのデータ構造を示した図

	サイズ	オフセット	内　　容	
	1ワード	0h	識別子　'MZ'	DOS EXE ヘッダの一部
			：	
	1ワード	18h	OS／2　EXE　識別子＝0x40	
			：	
	1ワード	3Ch	OS／2　EXEへのオフセット	

	サイズ	オフセット	内　　容	
	1ワード	0h	識別子　'NE'	OS／2 EXE ヘッダの一部
	1バイト	2h	リンカのバージョン番号	
	1バイト	3h	リンカのリビジョン番号	
●	1ワード	4h	エキスポートエントリテーブルのO2／2　EXE相対オフセット	
●	1ワード	6h	エキスポートエントリテーブルバイトサイズ	
	2ワード	8h	CRC－32	
	1ワード	Ch	フラグ値 8xxxh　→　DLL 00xxh　→　Not Specified 01xxh　→　Not Window Compat（フルスクリーン専用） 02xxh　→　Window Compat（PMセッションで動作可） 03xxh　→　Window API（PMのAT）	
	1ワード	Eh	自動データセグメントのセグメント番号	
	1ワード	10h	ローカルヒープサイズ	
	1ワード	12h	スタックサイズ	
	2ワード	14h	CS：IPアドレス	
	2ワード	18h	SS：SPアドレス	
●	1ワード	1Ch	セグメントテーブルの項目数	
	1ワード	1Eh	モジュール参照テーブルのエントリ数	
●	1ワード	20h	非常駐ネームテーブルのバイトサイズ	
●	1ワード	22h	セグメントテーブルのOS／2　EXE相対オフセット	
●	1ワード	24h	リソーステーブルのOS／2　EXE相対オフセット	
●	1ワード	26h	常駐ネームテーブルのOS／2　EXE相対オフセット	
●	1ワード	28h	モジュール参照テーブルのOS／2　EXE相対オフセット	
●	1ワード	2Ah	インポートネームテーブルのOS／2　EXE相対オフセット	
●	2ワード	2Ch	非常駐ネームテーブルのファイルの先頭からのオフセット	
	1ワード	30h	移動可能エントリの数	
	1ワード	32h	セグメント境界シフトカウント（2^ のx値）	
●	1ワード	34h	リソーテーブルエントリ数	
	1バイト	36h	OSタイプ（1＝OS／2）	

※　●のついている値は、旧ファイルと新ファイルのものをそれぞれ変数内に保持する。
※　37h以降は、Windowsのみで使用されているエリアであり、OS／2では使用されない。
　　従って、単純比較により差分を抽出する。

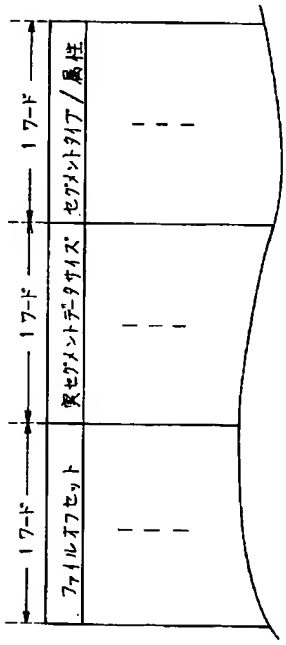
【図6】

OS/2 EXEヘッダ情報を示した図

変数	データの内容
entrytbl_off	エキスポートエントリテーブルのOS/2 EXE相対オフセット
entrytbl_len	エキスポートエントリテーブルバイトサイズ
segtbl_cnt	セグメントテーブルの項目数
segtbl_off	セグメントテーブルのOS/2 EXE相対オフセット
resnamtbl_off	常駐ネームテーブルのOS/2 EXE相対オフセット
modref_off	モジュール参照テーブルのOS/2 EXE相対オフセット
importnam_off	インポートネームテーブルのOS/2 EXE相対オフセット
nonresnam_len	非常駐ネームテーブルの長さ
nonresnam_off	非常駐ネームテーブルのファイル先頭からのオフセット
rsrctbl_off	リソーステーブルのOS/2 EXE相対オフセット
rsrctbl_cnt	リソーステーブルエントリ数

【図7】

セグメントテーブル部のデータ構造を示した図



【図9】

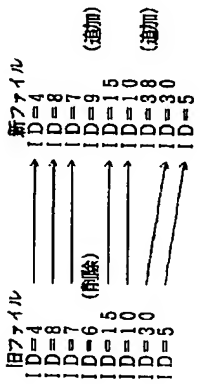
インポートネームテーブル部のデータ構造を示した図

サイズ	オフセット	内容
1バイト	0h	文字列のバイト長 0=テーブルエンド
nバイト	1h	名前列

【図11】

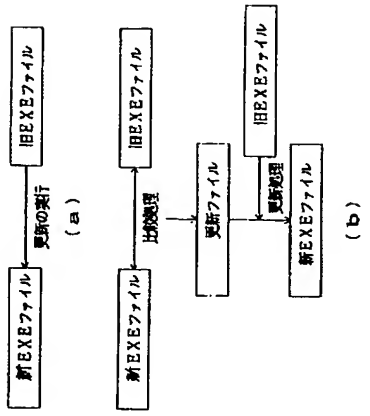
【図18】

リソースの対応関係の説明図



【図28】

ファイル更新の説明図

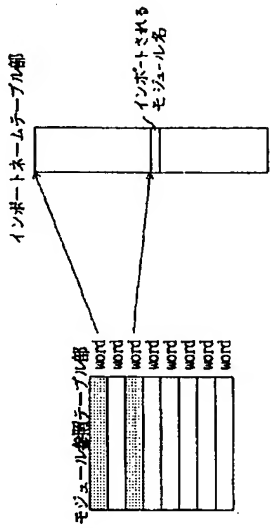


リソーステーブル部のデータ構造を示した図

サイズ	オフセット	内容
1ワード	0h	リソースのタイプID
1ワード	2h	リソースのネームID

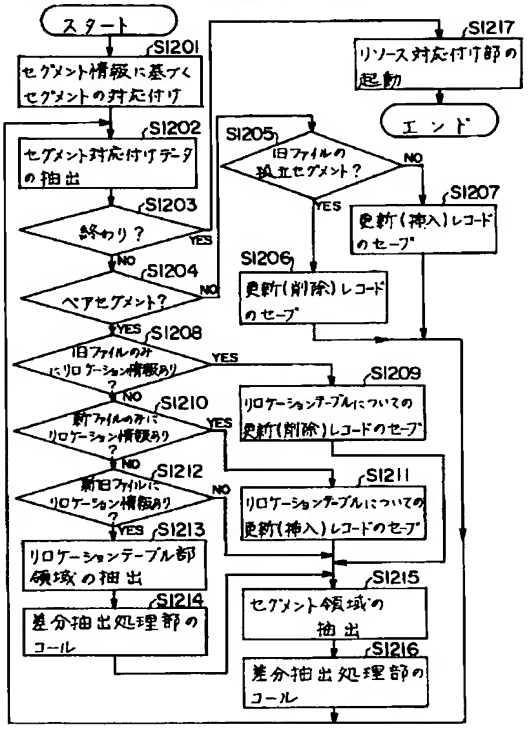
【図１０】

モジュール参照テーブル部とインポートネームテーブル部の関係を示した図



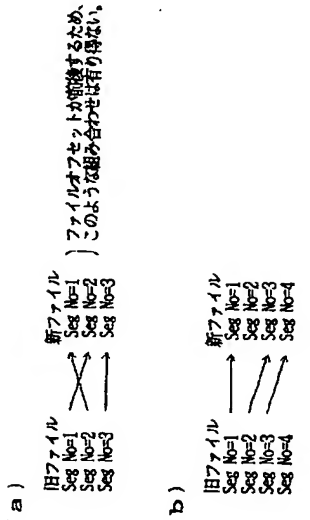
【図１２】

セグメントデータ対応付け部の動作フローチャート



【図１４】

セグメントの対応関係の説明図



【図２７】

更新ファイルのデータ構造を示した図

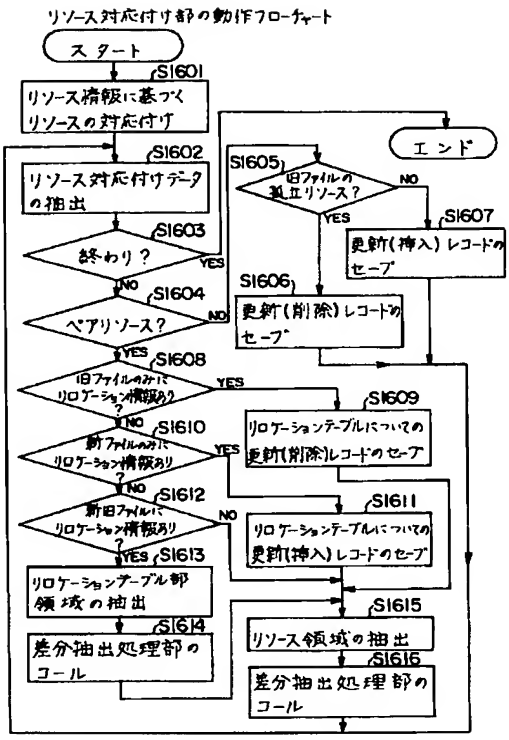
(a) ファイル構造

RecordID	DataOffset	DataLength	Data[n]
⋮	⋮	⋮	⋮

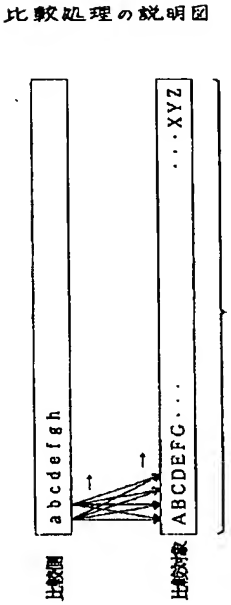
(b) RecordIDの意味

RecordID	意味	備考
0	置換レコード	DataLengthは、置換/挿入するデータ長を表す。
1	挿入レコード	
2	削除レコード	この場合、DataLength 何バイト削除するのかを表し、Dataは存在しない

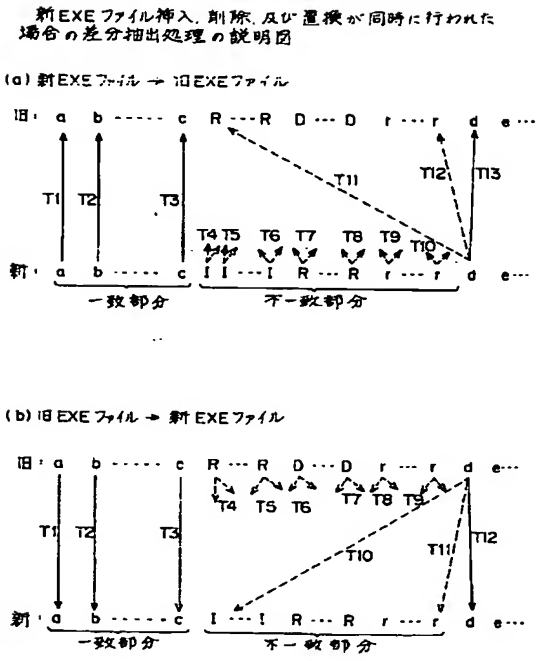
【図１６】


リソース対応付け部の動作フローチャート
スタート
S1601
リソース情報に基づくリソースの対応付け
S1602
リソース対応付けデータの抽出
S1603
終わり？
YES
NO
S1604
ペアリソース？
NO
YES
S1608
旧ファイルのみにリロケーション情報あり？
YES
NO
S1609
リロケーションテーブルについての更新(削除)レコードのセーブ
S1610
新ファイルのみにリロケーション情報あり？
YES
NO
S1611
リロケーションテーブルについての更新(挿入)レコードのセーブ
S1612
新旧ファイルにリロケーション情報あり？
NO
YES
S1613
リロケーションテーブル部領域の抽出
S1614
差分抽出処理部のコール
S1615
リソース領域の抽出
S1616
差分抽出処理部のコール
エンド
S1605
旧ファイルの孤立リソース？
NO
YES
S1607
更新(挿入)レコードのセーブ
S1606
更新(削除)レコードのセーブ

【図２１】

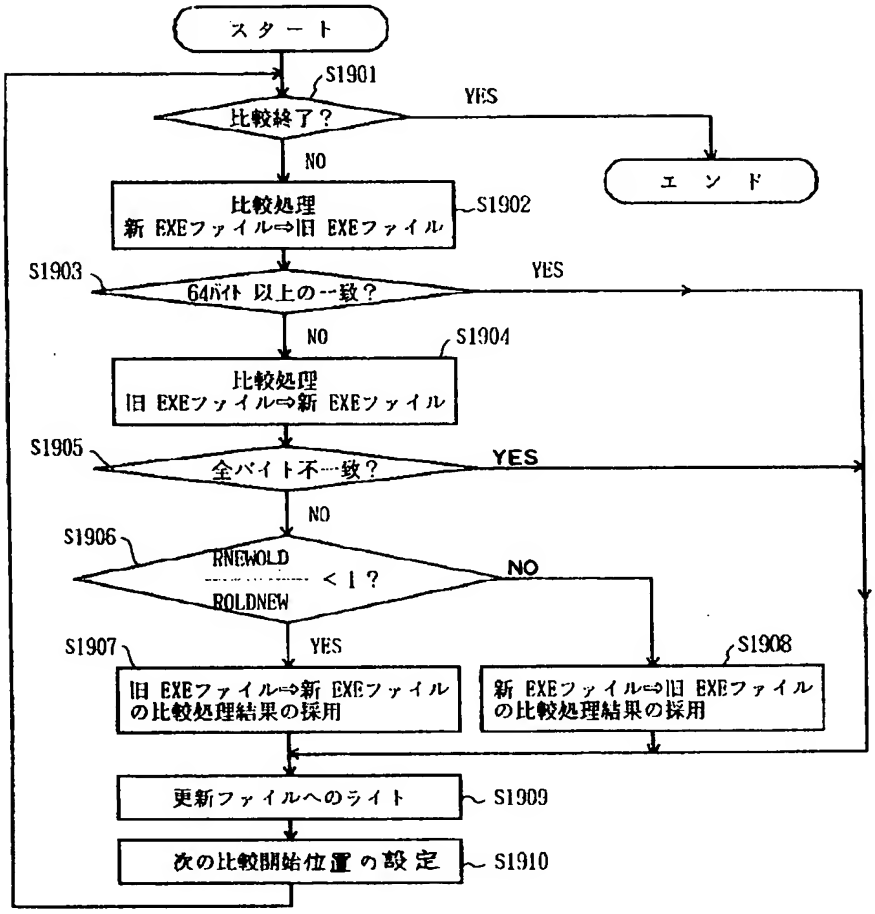

比較処理の説明図
abcdefgh
ABCDEFG・・・
・・・XYZ
比較元
比較対象

【図２５】


新EXEファイル挿入，削除，及び置換が同時に行われた場合の差分抽出処理の説明図
(a) 新EXEファイル → 旧EXEファイル
旧：
新：
T1
T2
T3
T4 T5
T6
T7
T8
T9
T10
T11
T12
T13
一致部分
不一致部分
(b) 旧EXEファイル → 新EXEファイル
旧：
新：
T1
T2
T3
T4 T5 T6
T7 T8 T9
T10
T11
T12
一致部分
不一致部分

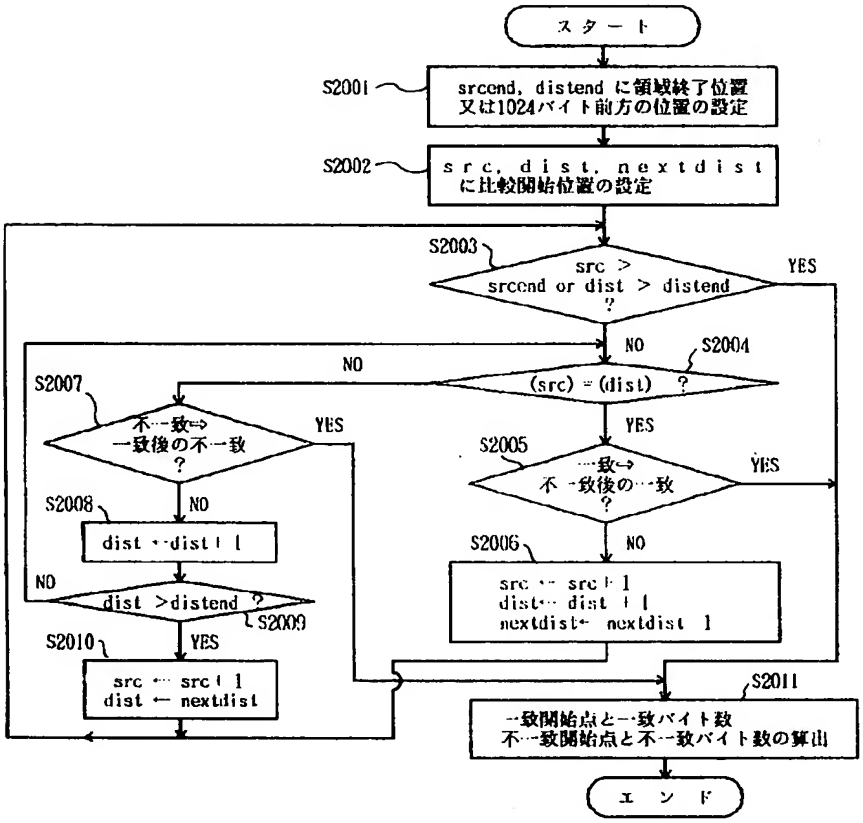
【図１９】

差分抽出処理の動作フローチャート



【図２０】

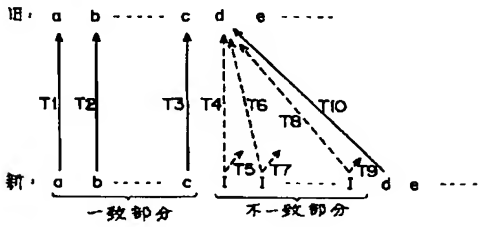
比較処理の動作フローチャート



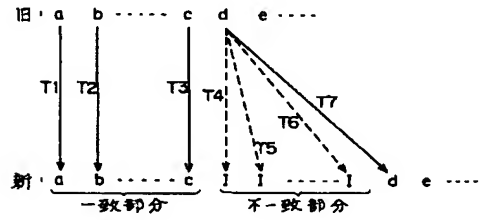
【図２２】

新EXEファイルに挿入が行われた場合の差分抽出処理の説明図

(a) 新EXEファイル → 旧EXEファイル



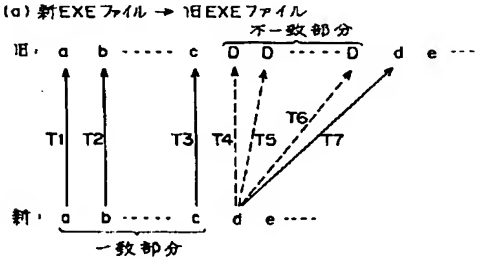
(b) 旧EXEファイル → 新EXEファイル



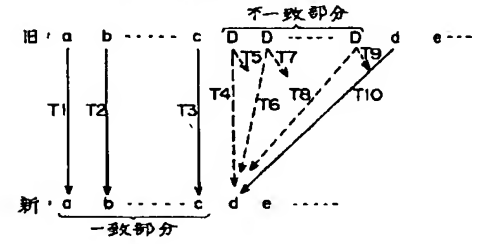
【図２３】

新EXEファイルで削除が行われた場合の差分抽出処理の説明図

(a) 新EXEファイル → 旧EXEファイル



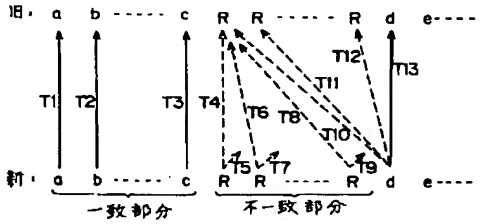
(b) 旧EXEファイル → 新EXEファイル



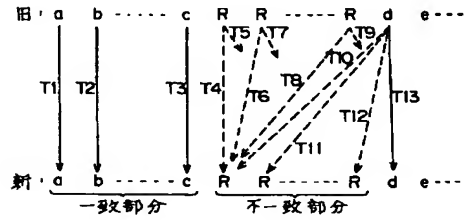
【図２４】

新EXEファイルで置換が行われた場合の差分抽出処理の説明図

(a) 新EXEファイル → 旧EXEファイル



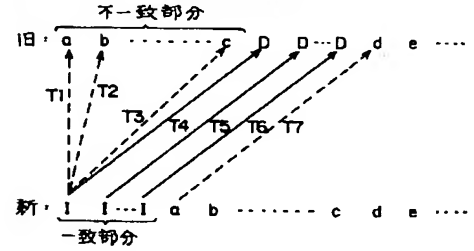
(b) 旧EXEファイル → 新EXEファイル



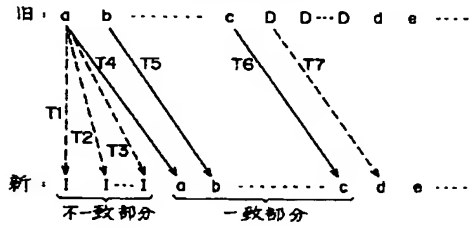
【図２６】

新EXEファイルで特殊な挿入及び削除が同時に行われた場合の差分抽出処理の説明図

(a) 新EXEファイル → 旧EXEファイル



(b) 旧EXEファイル → 新EXEファイル



【図29】

ファイル更新環境の説明図